TOWARD GROUNDED SPATIO-TEMPORAL REASONING

A Thesis
Presented to
The Academic Faculty

By

Chih-Yao Ma

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2020

TOWARD GROUNDED SPATIO-TEMPORAL REASONING

Approved by:

Dr. Ghassan AlRegib, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Dr. Patricio Antonio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology

Dr. Marcus Rohrbach
Facebook AI Research

Date Approved: March 9, 2020

I dedicate this thesis to,

My Mother Li-Yun,

My Father Chien-Pao,

My Brother Chih-Shun,

ACKNOWLEDGEMENTS

I am truly grateful to Prof. Ghassan AlRegib for giving me an opportunity to come to Georgia Tech and work with him in 2014. He offered this great opportunity without knowing me beforehand, and this opportunity has certainly change my life thereafter. I have certainly learned a lot from him during this journey both on technical knowledge and career development. His guidance through out my PhD journey has been a great help, and I would not have made this far without it.

I would also like to express my sincere thanks to Prof. Zsolt Kira, who has been frequently providing research feedback that made many of my research publications possible. His continual supports certainly also plays a key factor to the deliveries and outcomes of my PhD. We started to collaborate in 2016 when he was at GTRI, and I am happy that we continued to work together after he officially joined Georgia Tech as an assistant professor.

I would like to thank Prof. Patricio Antonio Vela, ..., for taking their precious time to serve on my dissertation committee. I want to thank all former and current members of the OLIVEs lab for being my colleagues and friends. I am especially thankful to: Can Temel, Mohammed Aabed, and Zhen Wang who guided me through the beginning of my PhD when I first joined the lab; Yazeed Alaudah, Motaz AlFarraj, Yuting Hu, Amir Shafiq, and Tariq Alshawi who have been great and fun colleagues to work with; Mohit Prabhushankar, Gukyeong Kwon, Charlie Lehman and Jinsol Lee, the new generation of the lab, who are passionate about research and made the lab space full of joy to sit in; Min-Hung Chen, who has a great colleague to work with and my dear friend after work even since I started my PhD journey in 2014.

I would also like to express my gratitude for my collaborators during my three internships: Asim Kadav, who was the mentor for my first internship at NEC Machine Learning and believed in me through out the entire internship. Jiasen Lu, Zuxuan Wu, Caiming Xiong, Richard Socher who collaborated with me during my internship at Salesforce Re-

search. Finally, Yannis Kalantidis, Marcus Rohrbach, Kan Chen, and Peter Vajda for supporting me and provided helpful feedback during my internship at Facebook.

Coming to Atlanta in 2014 was a big change for me. I did not know anyone locally and now I have a group of friends who are like a second family to me. I want to thank them for being my dearest friends and accompanies through the ups and downs of my PhD journey: Jonathan, Vivi, Jim, Brian, Enzo, Alice, Miki, Carol, and Christina.

Lastly, I would like to thank my family for being the biggest supports through my PhD journey. My partner, Grace Hsu, for bring joys and colored my life in addition to her warmhearted and unconditional supports. My mother, Li-Yun, who made who I am today. She has been open-minded and supporting all decisions I made even since I decided to pursue my PhD degree in the U.S. My dad, Chien-Pao, who frequently check-in with me and want me to be nothing but happy and healthy for my life. My twin brother, Chih-Shun, who knows me the best and thus can constantly guide me through many difficult decisions that I had to make over the past couple of years.

This past couple of years is certainly quite a journey for me. I have been through many ups and downs in exams, research projects, publications. At some points, I even stopped believed in myself that I would be able to publish good papers and graduate with my PhD degree. In retrospective, I really couldn't have done it without all the supports and guidance from my advisor, colleagues, collaborators, friends, and family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

To develop an Artificial Intelligence (AI) system that can understand the world around us, it needs to be able to interpret and reason about the world we see and the language we speak. In recent years, there has been a lot of attention to research at the intersection of vision, temporal reasoning, and language. One of the major challenges is how to ensure proper *grounding* and perform reasoning across multiple modalities given the heterogeneity resides in the data when there is no or weak supervision of the data. For example, (1) in Vision-and-Language Navigation, how to ensure the navigation agent to identify which part of the instruction has been completed or ongoing and which part is potentially needed for the next action selection, and how to identify which direction to go by finding the part of the instruction that corresponds to the observed images. (2) in visual understanding, how to efficiently leverage object-level features for downstream visual understanding tasks like action recognition and visual captioning, how to detect interactions/relationships when there is no or weak supervision from classification labels or ground-truth image/video descriptions.

In my thesis, the goal is to leverage spatial, temporal, and language inputs for both visual and textual understanding. I showed (1) how to equip the concept of self-monitoring to a seq-to-seq model in order to develop a visual-textual co-grounded navigation agent that can follow human commands in natural language format, (2) how to introduce the *roll-back* concept to the seq-to-seq based navigation agent by leveraging the self-monitoring mechanism that we proposed, (3) how to efficiently achieve object-level fine-grained video understanding for both human action recognition and video captioning, and (4) how to enforce the visual captioning models to generate *grounded* descriptions via a novel cyclical training regimen without ground-truth grounding annotations and without adding extra computation during inference.

CHAPTER 1
INTRODUCTION

The world around us is highly structured. This structure manifests itself in the spatiotemporal data that captures the world around us, and in the natural language that describes it. To develop an Artificial Intelligence (AI) system that can understand the world around us, it needs to be able to interpret and reason about the world we see and the language we speak. A machine or a system that possess the ability to learn and inference from both vision and language modalities offer the unlimited possibility to advance in-depth understanding of the natural world and provide helps in many real-world scenarios, for instance, in-house robot assistant, large-scale video analysis system, etc.

Recent years, there have been a lot of attention on research at the intersection of vision, temporal reasoning, and language, *e.g.,* vision-and-language navigation (VLN) [1, 2, 3], video classification and captioning [4, 5, 6], temporal localization [7, 8], etc. One of the major challenges is how to ensure proper *grounding* and perform reasoning across multiple modalities given the heterogeneity resides in the data when there is no or weak supervision of the data. For example, (1) in VLN, how to ensure the navigation agent, which follows navigational instructions, to identify which part of the instruction has been completed or ongoing and which part is potentially needed for the next action selection, how to identify which direction to go by finding the part of the instruction that corresponds to the observed images, and identify which direction to go by finding the part of the instruction that corresponds to the observed surrounding images. (2) in image/video understanding, how to efficiently leverage object-level features for downstream visual understanding tasks like action recognition and visual captioning, how to detect interactions/relationships when there is no or weak supervision from classification labels or ground-truth image/video descriptions, and how to detect and reason beyond pairwise interactions.

In my thesis, the goal is to leverage spatial, temporal, and language inputs for both visual and textual understanding that aims at (1) developing navigation agents that can follow natural language commands given by human, and (2) developing systems that can automatically analyze large-scale visual data and describe the content of the image/video. Specifically, for navigation agents, I showed (1) how to equip the concept of *self-monitoring* to a seq-to-seq model in order to develop a visual-textual co-grounded navigation agent that can follow human commands, and (2) how to introduce the *regretful* concept to the seq-to-seq based navigation agent by leveraging the self-monitoring capability so that the agent can perform a local search on a navigation graph. For systems that can analyze and describe images and videos, (1) I first showed how to efficiently achieve object-level fine-grained video understanding for both human action recognition and video captioning tasks by leveraging the structured visual relationships resides in spatiotemporal data. (2) I then demonstrate how to enforce the visual captioning models to generate *grounded* descriptions for the visual content via an novel cyclical training regimen.

1.1 Vision-and-Language Navigation Agent

Enabling robots to autonomously navigate in complex environments has attracted considerable attention for decades [9, 10, 11, 12]. Key to a successful navigation system are sensors that provide position information and maps that enable route planning based on the destination. However, the construction of maps and location-aware sensors like GPS limit the use of such systems in novel and unseen environments. Instead, Vision-and-Language navigation (VLN) [13], requiring the agent to follow natural language descriptions to navigate in photo-realistic unknown environments, offers an easier way to transfer across environments. However, this is a particularly challenging problem, since it requires the agent to effectively analyze both visual and textual signals, functioning as a route planner in lieu of location sensors and maps, and to determine its current state and the distance to the goal.

In Chapter 3, we first introduce a *self-monitoring* agent with two complementary com-

ponents: (1) **visual-textual co-grounding module** to locate the instruction completed in the past, the instruction required for the next action, and the next moving direction from surrounding images and (2) **progress monitor** to ensure the grounded instruction correctly reflects the navigation progress. We test our self-monitoring agent on a standard benchmark and analyze our proposed approach through a series of ablation studies that elucidate the contributions of the primary components.

Built upon the *self-monitoring* agent and inspired by the intuition of viewing the problem as a search on a navigation graph, we then propose to use the progress monitor as a learnable heuristic for search. Specifically, two modules are proposed and incorporated into an end-to-end *regretful* agent: 1) A learned mechanism to perform backtracking, which decides whether to continue moving forward or roll back to a previous state — **Regret Module**, and 2) A mechanism to help the agent decide which direction to go next by showing directions that are visited and their associated progress estimate — **Progress Marker**. Combined, the proposed approach significantly outperforms current state-of-the-art methods.

1.2 Object-Level Fine-grained Video Understanding

Video understanding tasks such as activity recognition and caption generation are crucial for various applications in surveillance, video retrieval, human behavior understanding, etc. Although recent state-of-the-art approaches for action recognition and video captioning have demonstrated significant improvements over standard datasets, they often focus on representing the overall visual scene (coarse-grained) as sequence of inputs that are combined with temporal pooling, *e.g.,* CRF, LSTM, 1D Convolution, attention, and NetVLAD [14, 4, 15, 16], or use 3D Convolution for the whole video sequence [17, 18, 19]. These approaches ignore the fine-grained details of the scene and do not infer interactions between various objects in the video.

In Chapter 5, we propose to efficiently learn higher-order interactions between arbi-

trary subgroups of objects for fine-grained video understanding. Specifically, we present a generic recurrent module — **Recurrent Higher-Order Interaction Module**, which dynamically discovers higher-order object interactions via an efficient dot-product attention mechanism combined with temporal reasoning. We demonstrate the proposed concept and associated approach on two mainstream video understanding tasks: action recognition and video captioning. To the best of our knowledge, this is the first work modeling object interactions and relationships on open domain large-scale video datasets.

1.3 Grounded Visual Captioning

Visual captioning is a challenging problem that involves generating a natural language sentence to accurately summarize the content of an image or video. However, image and video captioning models are frequently not well grounded [20], which often resulting model bias [21] and hallucination of objects [22]. This makes captioning models less reliable and trustworthy, which is essential if we hope that such models will have a significant practical impact in the real world, *e.g.,* assisting people in need [23].

In this thesis, I demonstrate how we can enforce the generated caption to be visually grounded via a proposed cyclical training regimen. Our proposed method is based on the one-to-one correspondence assumption, where each visually groundable word describes one particular region(s), and each particular region(s) corresponds to one visually groundable word, which holds in current benchmark datasets and most of the real-world cases. Specifically, we decompose the grounded visual captioning problem into a pair of dual tasks: visual captioning and visually groundable word localization. During training, the decoder described in 5.2 is first used for generating descriptions for image or video. We then leverage a simple **localizer** that maps each word in a generated caption to region(s) in an image or video. Finally, the decoder is required to reconstruct the ground-truth caption using the localized region(s). This training regimen can regularize the decoder to generate words that can be located to region(s) in the image by the localizer and being regenerated

by the same decoder, therefore, leading to a visually grounded decoder. Our proposed framework only requires learning one extra fully-connected layer (the localizer), a layer that can be removed at test time. We show that our model significantly improves grounding accuracy without relying on grounding supervision or introducing extra computation during inference for both image and video captioning tasks.

CHAPTER 2
LITERATURE SURVEY

2.1 Vision-and-Language Navigation

2.1.1 Vision, Language, and Navigation

There is a plethora work investigating the combination of vision and language for a multitude of applications [24, 25, 26, 27, 28], etc. While success has been achieved in these tasks to handle massive corpora of *static* visual input and text data, a resurgence of interest focuses on equipping an agent with the ability to interact with its surrounding environment for a particular goal such as object manipulation with instructions [29, 30], grounded language acquisition [31, 32, 33, 34], embodied question answering [35, 36], and navigation [37, 38, 39, 10, 40, 41, 42, 43, 44, 11, 12, 45]. In this work, we concentrate on the recently proposed the Vision-and-Language Navigation task [13]—asking an agent to carry out sophisticated natural-language instructions in a 3D environment. This task has application to fields such as robotics; in contrast to traditional map-based navigation systems, navigation with instructions provides a flexible way to generalize across different environments.

A few approaches have been proposed for the VLN task. For example, [13] address the task in the form of a sequence-to-sequence translation model. [46] introduce a guided feature transformation for textual grounding. [3] present a planned-head module by combing model-free and model-based reinforcement learning approaches. Recently, [2] propose to train a *speaker* to synthesize new instructions for data augmentation and further use it for pragmatic inference to rank the candidate routes. These approaches leverage attentional mechanisms to select related words from a given instruction when choosing an action, but those agents are deployed to explore the environment without knowing about what progress has been made and how far away the goal is. In this thesis, we propose a self-monitoring

agent that performs co-grounding on both visual and textual inputs and constantly monitors its own progress toward the goal as a way of regularizing the textual grounding.

2.1.2 Navigation and Learned Heuristics

Several works in vision and robotics have explored the intersection of learning and planning. In robotics, planning systems must often explore large search trees for getting from start to goal, and selection of the next state to expand must be done intelligently to reduce computation. Often fixed heuristics (*e.g.,* distance to goal) are used, but these are static, require known goal locations, and are used for optimal A*-style algorithms rather than greedy best-first search, which is what can be employed on robots when maps are not available [47]. Recently, several learning-based approaches have been developed for such heuristics, including older works that learn residuals for existing heuristics [48], heuristic ranking methods that enable refinement of new ones [49] as well as learning of a heuristic policy in a Markov Decision Process (MDP) formulation to directly optimize search effort by taking into account history and contextual information [50]. In our work, we similarly learn to estimate a heuristic (progress monitor) and use it for action selection, showing that the resulting estimates can generalize to unseen environments. We also develop an architecture to explicitly learn when to backtrack based on this progress monitor (with a Progress Marker to reduce the chance of choosing the same action again after backtracking unless warranted), which further improves navigation performance.

2.1.3 Modern Reinforcement Learning.

Modern Reinforcement Learning (RL) methods like Advantage Actor Critic (A2C) or Asynchronous Advantage Actor Critic (A3C) [51] methods are related to both our proposed Self-Monitoring agent [1] and the proposed Regretful agent [52]. Specifically, the progress monitor in the Self-Monitoring agent is similar to the value function in RL, and the difference between progress marker of a viewpoint and current progress estimation (denote

as $\Delta \boldsymbol{v}_{t,k}^{marker}$, see Sec. 4.1.2) is conceptually similar to the advantage function. However, the advantage function in RL serves as a way to regularize and improve the training of the policy network. We instead associate the $\Delta \boldsymbol{v}_{t,k}^{marker}$ directly to all navigable states, and this $\Delta \boldsymbol{v}_{t,k}^{marker}$ has a direct impact on the agent deciding next action even during inference. While having an accurate value estimate for VLN with dynamic and implicit goals may reduce the need for this formulation, we however believe that this is hardly possible because of the lack of training data. On the other hand, relating to the proposed end-to-end learned regret module, *Leave no Trace* [53] learns a forward and a *reset* policy to reset the environment for preventing the policy entering a non-reversible state. Instead of learning to reset, we learn to rollback to a previous state and continue the navigation task with a policy network that learns to decide a better next step.

2.2 Fine-grained Spatiotemporal Video Understanding

We discuss existing work on video understanding based on action recognition and video captioning as well as related work on detecting visual relationships in images and videos.

2.2.1 Action Recognition

Recent work on action recognition using deep learning involves learning compact (coarse) representations over time and use pooling or other aggregation methods to combine information from each video frame, or even across different modalities [54, 55, 15, 16, 56]. The representations are commonly obtained directly from forward passing a single video frame or a short video snippet to a 2D ConvNet or 3D ConvNet [17, 18, 19]. Another branch of work uses Region Proposal Networks (RPNs) to jointly train action detection models [57, 58, 59]. These methods use an RPN to extract object features (ROIs), but they do not model or learn interactions between objects in the scene. Distinct from these models, we explore human action recognition task using coarse-grained context information and fine-grained higher-order object interactions. Note that we focus on modeling object interactions for un-

derstanding video in a fine-grained manner and we consider other modalities, *e.g.,* optical flow and audio information, to be complementary to our method.

2.2.2 Video Captioning

Similar to other video tasks using deep learning, initial work on video captioning learn compact representations combined over time. This single representation is then used as input to a decoder, *e.g.,* LSTM, at the beginning or at each word generation to generate a caption for the target video [60, 61, 62]. Other work additionally uses spatial and temporal attention mechanisms to selectively focus on visual content in different space and time during caption generation [63, 64, 65, 66, 67]. Similar to using spatial attention during caption generation, another line of work has additionally incorporated semantic attributes [68, 69, 70, 71]. However, these semantic or attribute detection methods, with or without attention mechanisms, do not consider object relationships and interactions, i.e. they treat the detected attributes as a bag of words. Our work, SINet-Caption uses higher-order object relationships and their interactions as visual cues for caption generation.

2.2.3 Interactions/Relationships in images

Recent advances in detecting visual relationships in images use separate branches in a ConvNet to explicitly model objects, humans, and their interactions [72, 73]. Visual relationships can also be realized by constructing a scene graph which uses a structured representation for describing object relationships and their attributes [74, 75, 76, 77]. Other work on detecting visual relationships explore relationships by pairing different objects in the scene [78, 79, 80, 81]. While these models can successfully detect visual relationships for images, a scene with many objects may have only a few individual interacting objects. It would be inefficient to detect all relationships across all individual object pairs [82], making these methods intractable for the video domain.

2.2.4 Interactions/Relationships in videos

Compared to the image domain, there is limited work in exploring relationships for video understanding. Ni et al. [83] use a probabilistic graphical model to track interactions, but their model is insufficient to model interactions involving multiple objects. To overcome this issue, Ni et al. [84] propose using a set of LSTM nodes to incrementally refine the object detections. In contrast, Lea et al. [85] propose to decompose the input image into several spatial units in a feature map, which then captures the object locations, states, and their relationships using shared ConvNets. However, due to lack of appropriate datasets, existing work focuses on indoor or cooking settings where the human subject along with the objects being manipulated are at the center of the image. Also, these methods only handle pairwise relationships between objects. However, human actions can be complex and often involve higher-order object interactions. Therefore, we propose to attentively model object inter-relationships and discover the higher-order interactions on large-scale and open domain videos for fine-grained understanding.

2.3 Grounded Visual Captioning

2.3.1 Visual captioning and grounding

Neural models for visual captioning have received significant attention recently [86, 5, 87, 88, 61, 89, 90, 91, 92, 93]. Most current state-of-the-art models contain attention mechanisms, allowing the process to focus on subsets of the image when generating the next word. These attention mechanisms can be defined over spatial locations [94], semantic metadata [95, 96, 97, 98] or a predefined set of regions extracted via a region proposal network [5, 99, 86, 87, 100, 101]. In the latter case, off-the-shelf object detectors are first used to extract object proposals [102, 103] and the captioning model then learns to dynamically attend over them when generating the caption.

Although attention mechanisms are generally shown to improve captioning quality and

metrics, it has also been shown that they don't really focus on the same regions as a human would [104]. This make models less trustworthy and interpretable, and therefore creating *grounded* image captioning models, *i.e.,* models that accurately link generated words or phrases to specific regions of the image, has recently been an active research area. A number of approaches have been proposed, *e.g.,* for grounding phrases or objects from image descriptions [105, 106, 107, 108, 109, 110], grounding visual explanations [111], visual co-reference resolution for actors in video [91], or improving grounding via human supervision [112]. Recently, [109] presented a model with self-attention based context encoding and direct grounding supervision that achieves state-of-the-art results in both the image and video tasks. They exploit ground-truth bounding box annotations to significantly improve the visual grounding accuracy. In contrast, we focus on reinforcing the visual grounding capability of the existing captioning model via a cyclical training regimen without using bounding box annotations and present a method that can increase grounding accuracy while maintaining comparable captioning performance with state of the arts.

2.3.2 Cyclical training

Cycle consistency [113, 114, 115, 116] has been used recently in a wide range of domains, including machine translation [115], unpaired image-to-image translation [114], visual question answering [117], question answering [118], image captioning [116], video captioning [119, 120], captioning and drawing [121] as well as domain adaptation [122]. While the cyclical training regime has been explored vastly in both vision and language domains, it has not yet been used for enforcing the *visual grounding* capability of a captioning model.

CHAPTER 3

SELF-MONITORING NAVIGATION AGENT FOR VISION-AND-LANGUAGE NAVIGATION

Recently, the Vision-and-Language (VLN) navigation task [13], which requires the agent to follow natural language instructions to navigate through a photo-realistic unknown environment, has received significant attention [3, 2]. In the VLN task, an agent is placed in an unknown realistic environment and is required to follow natural language instructions to navigate from its starting location to a target location. In contrast to some existing navigation tasks [9, 10, 11, 12], we address the class of tasks where the agent does not have an explicit representation of the target (*e.g.,* location in a map or image representation of the goal) to know if the goal has been reached or not [37, 38, 39, 30]. Instead, the agent needs to be aware of its navigation status through the association between the sequence of observed visual inputs to instructions.

Consider an example as shown in Fig. 3.1, given the instruction "*Exit the bedroom and go towards the table. Go to the stairs on the left of the couch. Wait on the third step.*", the agent first needs to locate which instruction is needed for the next movement, which in turn requires the agent to be aware of (*i.e.,* to explicitly represent or have an attentional focus on) which instructions were completed or ongoing in the previous steps. For instance, the action *"Go to the stairs"* should be carried out once the agent has exited the room and moved towards the table. However, there exists inherent ambiguity for "*go towards the table*". Intuitively, the agent is expected to *"Go to the stairs"* after completing "*go towards the table*". But, it is not clear what defines the completeness of *"Go towards the table"*. The completeness of an ongoing action often depends on the availability of the next action. Since the transition between past and next part of the instructions is a *soft boundary*, in order to determine when to transit and to follow the instruction correctly the agent is

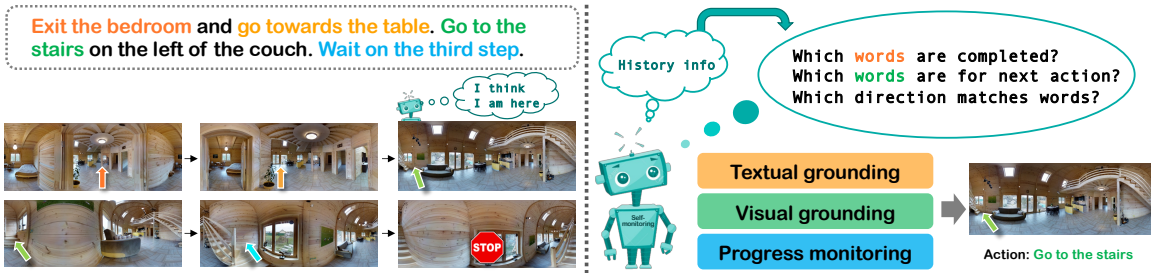

Figure 3.1: Vision-and-Language Navigation task and our proposed self-monitoring agent. The agent is constantly aware of what was completed, what is next, and where to go, as it navigates through unknown environments by following navigational instructions.

required to keep track of both grounded instructions. On the other hand, assessing the progress made towards the goal has indeed been shown to be important for goal-directed tasks in humans decision-making [123, 124, 125]. While a number of approaches have been proposed for VLN [13, 3, 2], previous approaches generally are not aware of which instruction is next nor progress towards the goal.

In this section, we propose an agent endowed with the following abilities: (1) identify which direction to go by finding the part of the instruction that corresponds to the observed images—*visual grounding*, (2) identify which part of the instruction has been completed or ongoing and which part is potentially needed for the next action selection—*textual grounding*, and (3) ensure that the grounded instruction can correctly be used to estimate the progress made towards the goal, and apply regularization to ensure this —*progress monitoring*. Therefore, we introduce the self-monitoring agent consisting of two complementary modules: **visual-textual co-grounding** and **progress monitor**.

More specifically, we achieve both visual and textual grounding simultaneously by incorporating the full history of grounded instruction, observed images, and selected actions into the agent. We leverage the structural bias between the words in instructions used for action selection and progress made towards the goal and propose a new objective function for the agent to measure how well it can estimate the completeness of instruction-following. We then demonstrate that by conditioning on the positions and weights of grounded instruc-

tion as input, the agent can be self-monitoring of its progress and further ensure that the textual grounding accurately reflects the progress made.

Overall, we propose a novel *self-monitoring* agent for VLN and make the following contributions: (1) We introduce the **visual-textual co-grounding** module, which performs grounding interdependently across both visual and textual modalities. We show that it can outperform the baseline method by a large margin. (2) We propose to equip the self-monitoring agent with a **progress monitor**, and for navigation tasks involving instructions instantiate this by introducing a new objective function for training. We demonstrate that, unlike the baseline method, the position of grounded instruction can follow both past and future instructions, thereby tracking progress to the goal. (3) With the proposed self-monitoring agent, we set the new state-of-the-art performance on both seen and unseen environments on the standard benchmark, with 8% absolute improvement in success rate on the unseen test set. Code is available at https://github.com/chihyaoma/selfmonitoring-agent.

3.1 Self-Monitoring Navigation Agent

3.1.1 Notation.

Given a natural language instruction with L words, its representation is denoted by $\boldsymbol{X} = \left\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L\right\}$, where $\boldsymbol{x}_l$ is the feature vector for the l-th word encoded by an LSTM language encoder. Following [2], we enable the agent with panoramic view. At each time step, the agent perceives a set of images at each viewpoint $\boldsymbol{v}_t = \left\{\boldsymbol{v}_{t,1}, \boldsymbol{v}_{t,2}, ..., \boldsymbol{v}_{t,K}\right\}$, where K is the maximum number of navigable directions[1], and $\boldsymbol{v}_{t,k}$ represents the image feature of direction k. The co-grounding feature of instruction and image are denoted as $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{v}}_t$ respectively. The selected action is denoted as $\mathbf{a}_t$. The learnable weights are denoted with $\boldsymbol{W}$, with appropriate sub/super-scripts as necessary. We omit the bias term $\boldsymbol{b}$ to avoid

[1]Empirically, we found that using only the images on navigable directions to be slightly better than using all 36 surrounding images (12 headings $\times$ 3 elevations with 30 degree intervals).

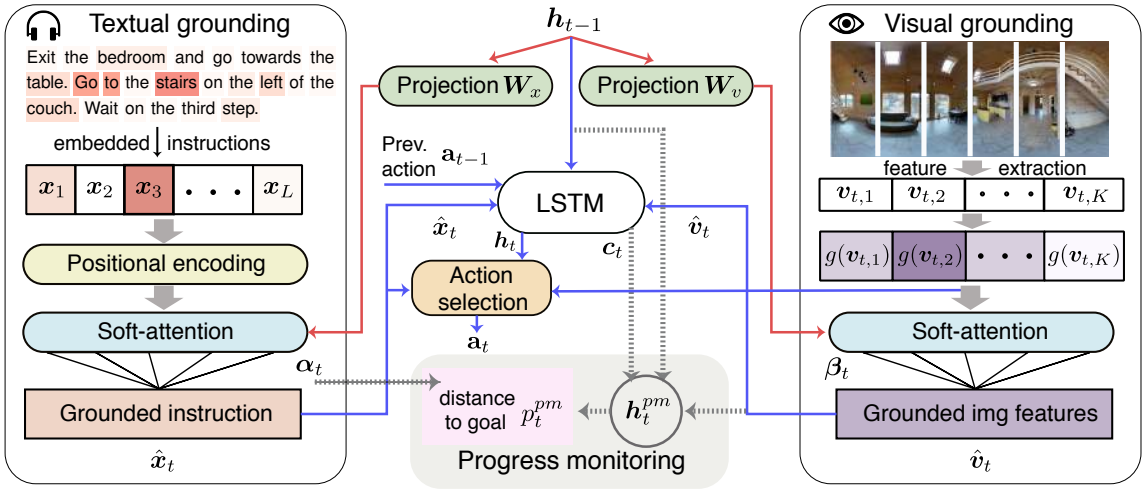

Figure 3.2: Proposed self-monitoring agent consisting of visual-textual co-grounding, progress monitoring, and action selection modules. *Textual grounding*: identify which part of the instruction has been completed or ongoing and which part is potentially needed for next action. *Visual grounding*: summarize the observed surrounding images. *Progress monitor*: regularize and ensure grounded instruction reflects progress towards the goal. *Action selection*: identify which direction to go.

notational clutter in the exposition.

3.1.2 Visual and Textual Co-Grounding

First, we propose a visual and textual co-grounding model for the vision and language navigation task, as illustrated in Fig. 3.2. We model the agent with a sequence-to-sequence architecture with attention by using a recurrent neural network. More specifically, we use Long Short Term Memory (LSTM) to carry the flow of information effectively. At each step t, the decoder observes representations of the current attended panoramic image feature $\hat{\boldsymbol{v}}_t$, previous selected action $\mathbf{a}_{t-1}$ and current grounded instruction feature $\hat{\boldsymbol{x}}_t$ as input, and outputs an encoder context $\boldsymbol{h}_t$:

$$\boldsymbol{h}_t = LSTM([\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{v}}_t, \mathbf{a}_{t-1}]) \tag{3.1}$$

where $[,]$ denotes concatenation. The previous encoder context $\boldsymbol{h}_{t-1}$ is used to obtain the textual grounding feature $\hat{\boldsymbol{x}}_t$ and visual grounding feature $\hat{\boldsymbol{v}}_t$, whereas we use current encoder context $\boldsymbol{h}_t$ to obtain next action $\mathbf{a}_t$.

Textual grounding. When the agent moves from one viewpoint to another, it is required to identify which direction to go by relying on a grounded instruction, i.e. which parts of the instruction should be used. To capture the relative position between words within an instruction, we incorporate the positional encoding $PE(\cdot)$ [126] into the instruction features. We then perform soft-attention on the instruction features $\boldsymbol{X}$, as shown on the left side of Fig. 3.2. The attention distribution over L words of the instructions is computed as:

$$z_{t,l}^{\text{textual}} = (\boldsymbol{W}_x \boldsymbol{h}_{t-1})^\top PE(\boldsymbol{x}_l), \quad \text{and} \quad \boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{z}_t^{\text{textual}}), \tag{3.2}$$

where $\boldsymbol{W}_x$ are parameters to be learnt. $z_{t,l}^{\text{textual}}$ is a scalar value computed as the correlation between word l of the instruction and previous hidden state $\boldsymbol{h}_{t-1}$, and $\boldsymbol{\alpha}_t$ is the attention weight over features in $\boldsymbol{X}$ at time t. Based on the textual attention distribution, the grounded textual feature $\hat{\boldsymbol{x}}_t$ can be obtained by $\hat{\boldsymbol{x}}_t = \boldsymbol{\alpha}_t^T \boldsymbol{X}$.

Visual grounding. In order to locate the completed or ongoing instruction, the agent needs to keep track of the sequence of images observed along the navigation trajectory. We thus perform visual attention over the surrounding views based on its previous hidden vector $\boldsymbol{h}_{t-1}$. The visual attention weight $\boldsymbol{\beta}_t$ can be obtained as:

$$z_{t,k}^{\text{visual}} = (\boldsymbol{W}_v \boldsymbol{h}_{t-1})^\top g(\boldsymbol{v}_{t,k}), \quad \text{and} \quad \boldsymbol{\beta}_t = \text{softmax}(\boldsymbol{z}_t^{\text{visual}}), \tag{3.3}$$

where g is a one-layer Multi-Layer Perceptron (MLP), $\boldsymbol{W}_v$ are parameters to be learnt. Similar to Eq. 3.2, the grounded visual feature $\hat{\boldsymbol{v}}_t$ can be obtained by the weighted sum over the visual features $\hat{\boldsymbol{v}}_t = \boldsymbol{\beta}_t^T \boldsymbol{V}$.

Action selection. To make a decision on which direction to go, the agent finds the image features on navigable directions with the highest correlation with the grounded nav-

igation instruction $\hat{\boldsymbol{x}}_t$ and the current hidden state $\boldsymbol{h}_t$. We use the inner-product to compute the correlation, and the probability of each navigable direction is then computed as:

$$o_{t,k} = (\boldsymbol{W}_a[\boldsymbol{h}_t, \hat{\boldsymbol{x}}_t])^\top g(\boldsymbol{v}_{t,k}) \quad \text{and} \quad \boldsymbol{p}_t = \text{softmax}(\boldsymbol{o}_t), \tag{3.4}$$

where $\boldsymbol{W}_a$ are the learnt parameters, $g(\cdot)$ is the same MLP as in Eq. 3.3, and $\boldsymbol{p}_t$ is the probability of each navigable direction at time t. We use categorical sampling during training to select the next action $\mathbf{a}_t$. Unlike the previous method with the panoramic view [2], which attends to instructions only based on the history of observed images, we achieve both textual and visual grounding using the shared hidden state output containing grounded information from both textual and visual modalities. During action selection, we rely on both hidden state output and grounded instruction, instead of only relying on grounded instruction.

3.1.3 Progress Monitor

It is imperative that the textual-grounding correctly reflects the progress towards the goal, since the agent can then implicitly know where it is now and what the next instruction to be completed will be. Thus, we propose to equip the agent with a **progress monitor** that serves as regularizer during training and prunes unfinished trajectories during inference.

Since the positions of localized instruction can be a strong indication of the navigation progress due to the structural alignment bias between navigation steps and instruction, the progress monitor can estimate how close the current viewpoint is to the final goal by conditioning on the positions and weights of grounded instruction. This can further enforce the result of textual-grounding to align with the progress made towards the goal and to ensure the correctness of the textual-grounding.

The progress monitor aims to estimate the navigation progress by conditioning on three inputs: the history of grounded images and instructions, the current observation of the

surrounding images, and the positions of grounded instructions. We therefore represent these inputs by using (1) the previous hidden state $\boldsymbol{h}_{t-1}$ and the current cell state $\boldsymbol{c}_t$ of the LSTM, (2) the grounded surrounding images $\hat{\boldsymbol{v}}_t$, and (3) the distribution of attention weights of textual-grounding $\boldsymbol{\alpha}_t$, as shown at the bottom of Fig. 3.2.

Our proposed progress monitor first computes an additional hidden state output $\boldsymbol{h}_t^{pm}$ by using grounded image representations $\hat{\boldsymbol{v}}_t$ as input, similar to how a regular LSTM computes hidden states except we use concatenation over element-wise addition for empirical reasons[2]. The hidden state output is then concatenated with the attention weights $\boldsymbol{\alpha}_t$ on textual-grounding to estimate how close the agent is to the goal[3]. The output of the progress monitor p_t^{pm}, which represents the completeness of instruction following, is computed as:

$$\boldsymbol{h}_t^{pm} = \sigma(\boldsymbol{W}_h([\boldsymbol{h}_{t-1}, \hat{\boldsymbol{v}}_t])) \otimes tanh(\boldsymbol{c}_t), \quad p_t^{pm} = tanh(\boldsymbol{W}_{pm}([\boldsymbol{\alpha}_t, \boldsymbol{h}_t^{pm}])) \tag{3.5}$$

where $\boldsymbol{W}_h$ and $\boldsymbol{W}_{pm}$ are the learnt parameters, c_t is the cell state of the LSTM, $\otimes$ denotes the element-wise product, and σ is the sigmoid function.

3.1.4 Training

We introduce a new objective function to train the proposed progress monitor. The training target y_t^{pm} is defined as the normalized distance in units of length from the current viewpoint to the goal, *i.e.,* the target will be 0 at the beginning and closer to 1 as the agent approaches the goal[4]. Note that the target can also be lower than 0, if the agent's current distance from the goal is farther than the starting point[5]. Finally, our self-monitoring agent is optimized with a cross-entropy loss and a mean squared error loss, computed with respect to the

[2] We found that using concatenation provides slightly better performance and stable training.

[3] We use zero-padding to handle instructions with various lengths.

[4] The target is set to 1 if the distance to the goal is less than 3.

[5] Alternatively, we found that setting the target to 0 when the current distance is farther than the starting point and using Sigmoid function achieve similar performance.

outputs from both action selection and progress monitor.

$$\mathcal{L}_{loss} = -\lambda \underbrace{\sum_{t=1}^{T} y_t^{nv} log(p_{k,t})}_{\text{action selection}} - (1-\lambda) \underbrace{\sum_{t=1}^{T} (y_t^{pm} - p_t^{pm})^2}_{\text{progress monitor}} \tag{3.6}$$

where $p_{k,t}$ is the action probability of each navigable direction, $\lambda = 0.5$ is the weight balancing the two losses, and y_t^{nv} is the ground-truth navigable direction at step t.

3.1.5 Inference

During inference, we follow [2] by using beam search. We propose that, while the agent decides which trajectories in the beams to keep, it is equally important to evaluate the state of the beams on actions as well as on the agent's confidence in completing the given instruction at each traversed viewpoint. We accomplish this idea by integrating the output of our progress monitor into the accumulated probability of beam search. At each step, when candidate trajectories compete based on accumulated probability, we integrate the estimated completeness of instruction-following p_t^{pm} (normalized between 0 to 1) with action probability $p_{k,t}$ to directly evaluate the partial and unfinished candidate routes: $p_t^{beam} = p_t^{pm} \times p_{k,t}$.

Without beam search, we use greedy decoding for action selection with one condition. If the progress monitor output decreases ($p_{t+1}^{pm} < p_t^{pm}$), the agent is required to move back to the previous viewpoint and select the action with next highest probability. We repeat this process until the selected action leads to increasing progress monitor output. We denote this procedure as *progress inference*.

3.2 Dataset and Implementation Details.

R2R Dataset. We use the Room-to-Room (R2R) dataset [13] for evaluating our proposed approach. The R2R dataset is built upon the Matterport3D dataset [127] and has 7,189 paths sampled from its navigation graphs. Each path has three ground-truth navigation

instructions written by humans. The whole dataset is divided into 4 sets: training, validation seen, validation unseen, and test sets unseen.

Evaluation metrics. (1) Navigation Error (NE), mean of the shortest path distance in meters between the agent's final position and the goal location. (2) Success Rate (SR), the percentage of final positions less than 3m away from the goal location. (3) Oracle Success Rate (OSR), the success rate if the agent can stop at the closest point to the goal along its trajectory. (4) Success rate weighted by (normalized inverse) Path Length (SPL) [128], the Success Rate trades-off against trajectory length.

3.2.1 Implementation details

We now discuss implementation details regarding input image features, network architecture, and the training procedure.

Image feature. Similar to previous work, we use the pre-trained ResNet-152 on ImageNet to extract image features. Each image feature is thus a 2048-d vector. The embedded feature vector for each navigable direction is obtained by concatenating an appearance feature with a 4-d orientation feature $[sin\phi; cos\phi; sin\theta; cos\theta]$, where ϕ and θ are the heading and elevation angles. Following the work in [2], the 4-dim orientation features are tiled 32 times, resulting a embedding feature vector with 2176 dimension.

Network architecture. The embedding dimension for encoding the navigation instruction is 256. We use a dropout layer with ratio 0.5 after the embedding layer. We then encode the instruction using a regular LSTM, and the hidden state is 512 dimensional. The MLP g used for projecting the raw image feature is $BN \rightarrow FC \rightarrow BN \rightarrow Dropout \rightarrow ReLU$. The FC layer projects the 2176-d input vector to a 1024-d vector, and the dropout ratio is set to be 0.5. The hidden state of the LSTM used for carrying the textual and visual information through time in Eq. 3.1 is 512. We set the maximum length of instruction to be 80, thus the dimension of the attention weights of textual grounding $\boldsymbol{\alpha}_t$ is also 80. The dimension of the learnable matrices from Eq. 3.2 to 3.5 are: $\boldsymbol{W}_x \in \mathbb{R}^{512\times512}$, $\boldsymbol{W}_v \in \mathbb{R}^{512\times1024}$,

Table 3.1: Performance comparison with the state of arts: Student-forcing [13], RPA [3], and Speaker-Follower [2]. *: with data augmentation. leaderboard: when using beam search, we modify our search procedure to comply with the leaderboard guidelines, i.e., all traversed viewpoints are recorded.

	Validation-Seen				Validation-Unseen				Test (unseen)			
Method	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Random	9.45	0.16	0.21	-	9.23	0.16	0.22	-	9.77	0.13	0.18	-
Student-forcing	6.01	0.39	0.53	-	7.81	0.22	0.28	-	7.85	0.20	0.27	-
RPA	5.56	0.43	0.53	-	7.65	0.25	0.32	-	7.53	0.25	0.33	-
Speaker-Follower	3.88	0.63	0.71	-	5.24	0.50	0.63	-	-	-	-	-
Speaker-Follower*	3.08	0.70	**0.78**	-	4.83	0.55	0.65	-	4.87	0.53	0.64	-
(leaderboard)	-	-	-	-	-	-	-	-	4.87	0.53	0.96	0.01
Ours (beam search)	3.23	0.70	**0.78**	0.66	5.04	0.57	**0.70**	0.51	4.99	0.57	0.68	0.51
(leaderboard)	-	-	-	-	-	-	-	-	4.99	0.57	0.95	0.02
Ours* (beam search)	**3.04**	**0.71**	**0.78**	**0.67**	**4.62**	**0.58**	0.68	**0.52**	**4.48**	**0.61**	0.70	**0.56**
(leaderboard)	-	-	-	-	-	-	-	-	**4.48**	**0.61**	**0.97**	0.02

$\boldsymbol{W}_a \in \mathbb{R}^{1024\times1024}$, $\boldsymbol{W}_h \in \mathbb{R}^{1536\times512}$, and $\boldsymbol{W}_{pm} \in \mathbb{R}^{592\times1}$.

Training. We use ADAM as the optimizer. The learning rate is $1e-4$ with batch size of 64 consistently through out all experiments. When using beam search, we set the beam size to be 15. We perform categorical sampling during training for action selection.

3.3 Experiments

3.3.1 Comparison with Prior Art

We first compare the proposed self-monitoring agent with existing approaches. As shown in Table 3.1, our method achieves significant performance improvement compared to the state of the arts without data augmentation. We achieve 70% SR on the seen environment and 57% on the unseen environment while the existing best performing method achieved 63% and 50% SR respectively. When trained with synthetic data[6], our approach achieves slightly better performance on the seen environments and significantly better performance on both the validation unseen environments and the test unseen environments when submitted to the test server. We achieve 3% and 8% improvement on SR on both validation and test unseen environments. Both results with or without data augmentation indicate that our proposed

[6]We use the exact same synthetic data generated from the Speaker as in [2] for comparison.

Table 3.2: Performance comparison with the state of arts *without beam search*: Student-forcing [13], RPA [3], and Speaker-Follower [2]. *: with data augmentation.

	Validation-Seen				Validation-Unseen				Test (unseen)			
Method	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Random	9.45	0.16	0.21	-	9.23	0.16	0.22	-	9.77	0.13	0.18	-
Student-forcing	6.01	0.39	0.53	-	7.81	0.22	0.28	-	7.85	0.20	0.27	-
RPA	5.56	0.43	0.53	-	7.65	0.25	0.32	-	7.53	0.25	0.33	-
Speaker-Follower*	3.36	0.66	0.74	-	6.62	0.36	0.45	-	6.62	0.35	0.44	0.28
Ours* (Greedy Decoding)	3.22	0.67	**0.78**	**0.58**	5.52	0.45	0.56	0.32	5.99	0.43	0.55	0.32
Ours* (Progress Inference)	**3.18**	**0.68**	0.77	**0.58**	**5.41**	**0.47**	**0.59**	**0.34**	**5.67**	**0.48**	**0.59**	**0.35**

approach is more generalizable to unseen environments.

Note that both Speaker-Follower and our approach in Table 3.1 use beam search. We additionally conduct experiments with *progress inference* (without beam search) introduced in Sec. 3.1.5. The results are shown in Table 3.2. We can see that our proposed method outperformed existing approaches with a large margin on both validation unseen and test sets. Our method with greedy decoding for action selection improved the SR by 9% and 8% on validation unseen and test set. When using progress inference for action selection, the performance on the test set significantly improved by 5% compared to using greedy decoding, yielding 13% improvement over the best existing approach.

3.3.2 Textually Grounded Agent

Intuitively, an instruction-following agent is required to strongly demonstrate the ability to correctly focus and follow the corresponding part of the instruction as it navigates through an environment.

We thus record the distribution of attention weights on instruction at each step as indications of which parts of the instruction being used for action selection. We average all runs across both validation seen and unseen dataset splits. Ideally, we expect to see the distribution of attention weights lies close to a diagonal, where at the beginning, the agent focuses on the beginning of the instruction and shifts its attention towards the end of instruction as it moves closer to the goal.

To demonstrate, we use the method with panoramic action space proposed in [2] as a

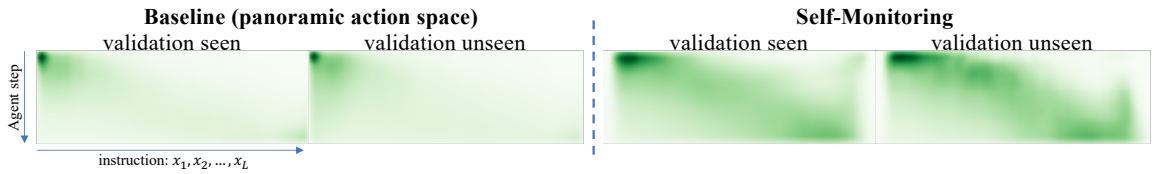

Figure 3.3: The positions and weights of grounded instructions as agents navigate by following instructions. Our agent with progress monitor demonstrates the grounded instruction used for action selection shifts gradually from the beginning of instructions towards the end. This is not true of the baseline method.

baseline for comparison. As shown in Figure 3.3, our self-monitoring agent with progress monitor demonstrates that the positions of grounded instruction over time form a line similar to a diagonal. This result may indicate that the agent successfully utilizes the attention on instruction to complete the task sequentially. We can also see that both agents were able to focus on the first part of the instruction at the beginning of navigation consistently. However, as the agent moves further in unknown environments, our self-monitoring agent can still successfully identify the parts of instruction that are potentially useful for action selection, whereas the baseline approach becomes uncertain about which part of the instruction should be used.

3.3.3 Ablation Study

We now discuss the importance of each component proposed in this work. We begin with the same baseline as before (agent with panoramic action space in [2])[7].

Co-grounding. When comparing the baseline with row #1 in our proposed method, we can see that our co-grounding agent outperformed the baseline with a large margin. This is due to the fact that we use the LSTM to carry both the textually and visually grounded content, and the decision on each navigable direction is predicted with both textually grounded instruction and the hidden state output of the LSTM. On the other hand, the baseline agent

[7]Note that our results for this baseline are slightly higher on val-seen and slightly lower on val-unseen than those reported, due to differences in hyper-parameter choices.

Table 3.3: Ablation study showing the effect of each proposed component. All methods use the panoramic action space. Note that, for methods using beam search during inference, only the last selected trajectory is used for evaluating OSR and SPL. *: we implemented the model from Speaker-Follower [2] with panoramic action space as baseline.

#	Co-Grounding	Progress Monitor	Inference Mode: Greedy Decoding	Inference Mode: Progress Inference	Inference Mode: Beam Search	Data Aug.	Validation-Seen NE ↓	Validation-Seen SR ↑	Validation-Seen OSR ↑	Validation-Seen SPL ↑	Validation-Unseen NE ↓	Validation-Unseen SR ↑	Validation-Unseen OSR ↑	Validation-Unseen SPL ↑
Baseline*							4.36	0.54	0.68	-	7.22	0.27	0.39	-
1	✓		✓				3.65	0.65	0.75	0.56	6.07	0.42	0.57	0.28
2	✓	✓	✓				3.72	0.63	0.75	0.56	5.98	0.44	0.58	0.30
3	✓	✓	✓			✓	3.22	0.67	**0.78**	0.58	5.52	0.45	0.56	0.32
4	✓	✓		✓			3.56	0.65	0.75	0.58	5.89	0.46	0.60	0.32
5	✓	✓		✓		✓	3.18	0.68	0.77	0.58	5.41	0.47	0.59	0.34
6	✓				✓		3.66	0.66	0.76	0.62	5.70	0.49	0.68	0.42
7	✓	✓			✓		3.23	0.70	**0.78**	0.66	5.04	0.57	**0.70**	0.51
8	✓	✓			✓	✓	**3.04**	**0.71**	**0.78**	**0.67**	**4.62**	**0.58**	0.68	**0.52**

relies on the LSTM to carry visually grounded content, and uses the hidden state output for predicting the textually grounded instruction. As a result, we observed that instead of predicting the instruction needed for selecting a navigable direction, the textually grounded instruction may match with the past sequence of observed images implicitly saved within the LSTM.

Progress monitor. Given the effective co-grounding, the proposed progress monitor further ensure that the grounded instruction correctly reflects the progress made toward the goal. This further improves the performance especially on the unseen environments as we can see from row #1 and #2.

When using the progress inference, the progress monitor serve as a progress indicator for the agent to decide when to move back to the last viewpoint. We can see from row #2 and #4 that the SR performance can be further improved around 2% on both seen and unseen environments.

Finally, we integrate the output of the progress monitor with the state-factored beam search [2], so that the candidate paths compete not only based on the probability of selecting a certain navigable direction but also on the estimated correspondence between the past trajectory and the instruction. As we can see by comparing row #2, #6, and #7, the progress monitor significantly improved the success rate on both seen and unseen environments and

is the key for surpassing the state of the arts even without data augmentation. We can also see that when using beam search without progress monitor, the SR on unseen improved 7% (row #1 vs #6), while using beam search integrated with progress estimation improved 13% (row #2 vs #7).

Data augmentation. In the above, we have shown each row in our approach contributes to the performance. Each of them increases the success rate and reduces the navigation error incrementally. By further combining them with the data augmentation pre-trained from the *speaker* [2], the SR and OSR are further increased, and the NE is also drastically reduced. Interestingly, the performance improvement introduced by data augmentation is smaller than from Speaker-Follower on the validation sets (see Table 3.1 for comparison). This demonstrates that our proposed method is more data-efficient.

3.4 Qualitative Results

We provide and discuss additional qualitative results on the *self-monitoring* agent navigating on seen and unseen environments. We first discuss four successful examples in Fig. 3.4 and 3.5, and followed by two failure examples in Fig. 3.6.

3.4.1 Successful Examples

In Fig. 3.4 (a), at the beginning, the agent mostly focuses on "*walk up*" for making the first movement. While the agent keeps its attention on "*walk up*" as completed instruction or ongoing action, it shifts the attention on instruction to "*turn right*" as it walks up the stairs. Once it reached the top of the stairs, it decides to turn right according to the grounded instruction. Once turned right, we can again see that the agent pays attention on both the past action "*turn right*" and next action "*walk straight to bedroom*". The agent continues to do so until it decides to stop by grounding on the word "*stop*".

In Fig. 3.4 (b), the agent starts by focusing on both "*enter bedroom from balcony*" and "*turn left*" to navigate. It correctly shifts the attention on textual grounding on the following

instruction. Interestingly, the given instruction "*walk straight across rug to room*" at step 3 is ambiguous since there are two rooms across the rug. Our agent decided to sneak out of the first room on the left and noticed that it does not match with the description from instruction. It then moved to another room across the rug and decided to stop because there is a rug inside the room as described.

In Fig. 3.5 (a), the given instruction is ambiguous as it only asks the agent to take actions around the stairs. Since there are multiple duplicated actions described in the instruction, e.g. "*walk up*" and "*turn left*", only an agent that is able to precisely follow the instruction step-by-step can successfully complete the task. Otherwise, the agent is likely to stop early before it reaches the goal. The agent also needs to demonstrate its ability to assess the completeness of instruction-following task in order to correctly stop at the right amount of repeated actions as described in the instruction.

In Fig. 3.5 (b), at the beginning (step 0), the agent only focuses on 'left' for making the first movement (the agent is originally facing the painting). We can see that at each step, the agent correctly focuses on parts of the instruction for making every movements, and it finally believes that the instruction is completed (attention on the last sentence period) and stopped.

3.4.2 Failure Examples

In Fig. 3.6 (a) step 1, although the attention on instruction correctly focused on "*take a left*" and "*go down*", the agent failed to follow the instruction and was not able to complete the task. We can however see that the progress monitor correctly reflected that the agent did not follow the given instruction successfully. The agent ended up stopping with progress monitor reporting that only 16% of the instruction was completed.

In Fig. 3.6 (b) step 2, the attention on instruction only focuses on "*go down*" and thus failed to associate the "*go down steps*" with the stairs previously mentioned in "*turn right to stairs*". The agent was however able to follow the rest of the instruction correctly by turning

right and stopping near a mirror. Note that, different from Fig. 3.6 (a), the final estimated completeness of instruction-following from progress monitor is much higher (16%), which indicates that the agent failed to be aware that it was not correctly following the instruction.

3.5 Summary

We introduce a novel *self-monitoring* agent which consists of two complementary modules: visual-textual co-grounding module and progress monitor. The visual-textual co-grounding module locates the instruction completed in the past, the instruction needed in the next action, and the moving direction from surrounding images. The progress monitor regularizes and ensures the grounded instruction correctly reflects the progress towards the goal by explicitly estimating the completeness of instruction-following. This estimation is conditioned on the positions and weights of grounded instruction. Our approach sets a new state-of-the-art performance on the standard Room-to-Room dataset on both seen and unseen environments. While we present one instantiation of self-monitoring for a decision-making agent, we believe that this concept can be applied to other domains as well.

(a) (b)

Figure 3.4: Successful self-monitoring agent navigates in two different unseen environments. Given the navigational instruction located at the top of the figure, the agent starts from starting position and follows the instruction towards the goal. The percentage of instruction completeness estimated by the proposed progress monitor gradually increases as the agent navigates and approaches the goal.

Figure 3.5: Successful self-monitoring agent navigates in (a) unseen and (b) seen environments. (a) The given instruction is ambiguous as it only asks the agent to take actions around the stairs. Since there are multiple duplicated actions described in the instruction, e.g. "*walk up*" and "*turn left*", only an agent that is able to precisely follow the instruction step-by-step can successfully complete the task. Otherwise, the agent is likely to stop early before it reaches the goal. (b) The agent correctly pays attention to parts of the instruction for making decisions on selecting navigable directions. Both the agents decide to stop when shifting the textual grounding on the last sentence period.

(a) (b)

Figure 3.6: Failed self-monitoring agent navigates in unseen environments. (a) The agent missed the "*take a left*" at step 1, and consequently unable to follow the following instruction correctly. However, note that the progress monitor correctly reflected that the instruction was not completed. When the agent decides to end the navigation, it reports that only 16% of the instruction was completed. (b) At step 2, the attention on instruction only focuses on "*go down*" and thus failed to associate the "*go down steps*" with the stairs previously mentioned in "*turn right to stairs*". The agent was however able to follow the rest of the instruction correctly by turning right and stopping near a mirror. Note that, different from (a), the final estimated completeness of instruction-following is much higher, which suggests that the agent failed to correctly be aware of its progress towards the goal.

CHAPTER 4
THE REGRETFUL NAVIGATION AGENT

Along with the *self-monitoring* navigation agent that we introduced in Chapter 3, dominant approaches in VLN frame the navigation task as a *sequence to sequence* problem [13]. For example, several enhancements such as synthetic data augmentation [2], pragmatic inference [2], and combinations of model-free and model-based reinforcement learning techniques [3] have also been proposed. These methods can be separated into two regimes: those that use beam search and obtain good success rate (with longer trajectory lengths) and those that use greedy action selection (and hence result in very low trajectory lengths) but obtain much lower success rates. In fact, there have recently been new metrics proposed that balance these two objectives [128]. Intuitively, the agent should perform intelligent action selection (akin to best-first search), without exhaustively exploring the search space. For robotics application, for example, the use of beam search is unrealistic as it would require the robot to explore a large number of possible trajectories.

In this chapter, we view the process of navigation as *graph search* across the navigation graph and employ two strategies, encoded within the neural network architecture, to enable navigation without the use of beam search. Specifically, we develop: 1) A **Regret Module** that provides a mechanism to allow the agent to *learn when to backtrack* [129, 130] and 2) We propose a **Progress Marker** mechanism that allows the agent to incorporate information from previous visits and reason about such visits and their associated progress estimates towards better action selection.

Specifically, in graph search a heuristic is used to make meaningful progress towards the goal in a manner that avoids exhaustive search but is more effective than naïve greedy search. We therefore build on recent work [1] that developed a *progress monitor* which is a learned mechanism that was used to estimate the progress made towards the goal (with low

values meaning progress has not been made and high values meaning the agent is closer to the goal). In that work, however, the focus was on the regularizing effect of the progress monitor as well as its use in beam search. Instead, we use this progress monitor effectively as a *learned heuristic* that can be used to determine directions that are more likely to lead towards the goal during inference.

We use the Progress Marker in two ways. First, we leverage the notion of backtracking, which is prevalent in graph search, by developing a *learned rollback mechanism* that decides whether to go back to the previous location or not (Regret Module). Second, we incorporate a mechanism to allow the agent to use the estimated progress it computed when visiting the viewpoints to choose the next action to perform after it has rolled back (Progress Marker). This allows the agent to know when particular directions have already been visited and the progress they resulted in, which can bias it to not re-visit states unless warranted. We do this by augmenting the visual state vectors with the progress estimates so that the agent can reduce the probability of revisiting such states (again, in a learned manner).

We demonstrate that these learned mechanisms are superior to greedy decoding. Our agent is able to achieve state-of-the-art results among published works both in terms of success rate (when beam search is not used) and more importantly the SPL [128] metric which incorporates path length, owing to our short trajectory lengths. In summary, our contributions include: 1) A graph search perspective on the instruction-based navigation problem, and use of a **learned heuristic** in the form of a progress monitor to effectively explore the navigation graph, 2) an end-to-end trainable **Regret Module** that can learn to decide when to roll back to the previous location given the history of textual and visual grounding observed, 3) a **Progress Marker** that can enable effective backtracking and reduce the probability of going to a visited location accordingly, and 4) state-of-the-art results on the VLN task. Our code is available at https://github.com/chihyaoma/regretful-agent.

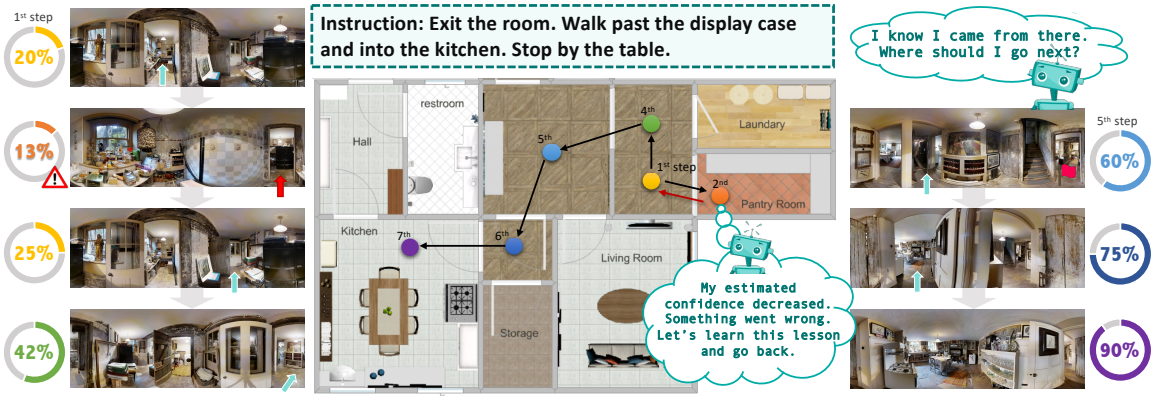

Figure 4.1: Vision-and-Language Navigation task and our proposed regretful navigation agent. The agent leverages the self-monitoring mechanism through time to decide when to roll back to a previous location and resume the instruction-following task.

4.1 Regretful Navigation Agent

The progress monitor previously mentioned reflects the agent's progress made towards the goal, and consequently its outputs will decrease or fluctuate if the agent selects an action leading to deviation from the goal. Conversely it will increase if it moves closer to the goal by completing the instruction. We posit that such a property, while conceptually simple, provides critical feedback for action selection. To this end, we leverage the outputs of the progress monitor to allow the agent to regret and backtrack using a **Regret Module** and a **Progress Marker**. (see Figure. 4.2). In particular, the Regret Module examines the progress made from the last step to the current step to decide whether to take a *forward* or *rollback* action. Once the agent regrets and rolls back to the previous location, the Progress Marker informs whether location(s) have been visited before and rates the visited location(s) according to the agent's confidence in completing the instruction-following task. Combining the two proposed methods, we show that the agent is able to perform a local search on the navigational graph by (1) assessing the current progress, (2) deciding when to roll back, and (3) selecting the next location after rollback occurs. In the following, we elaborate these two components in detail.

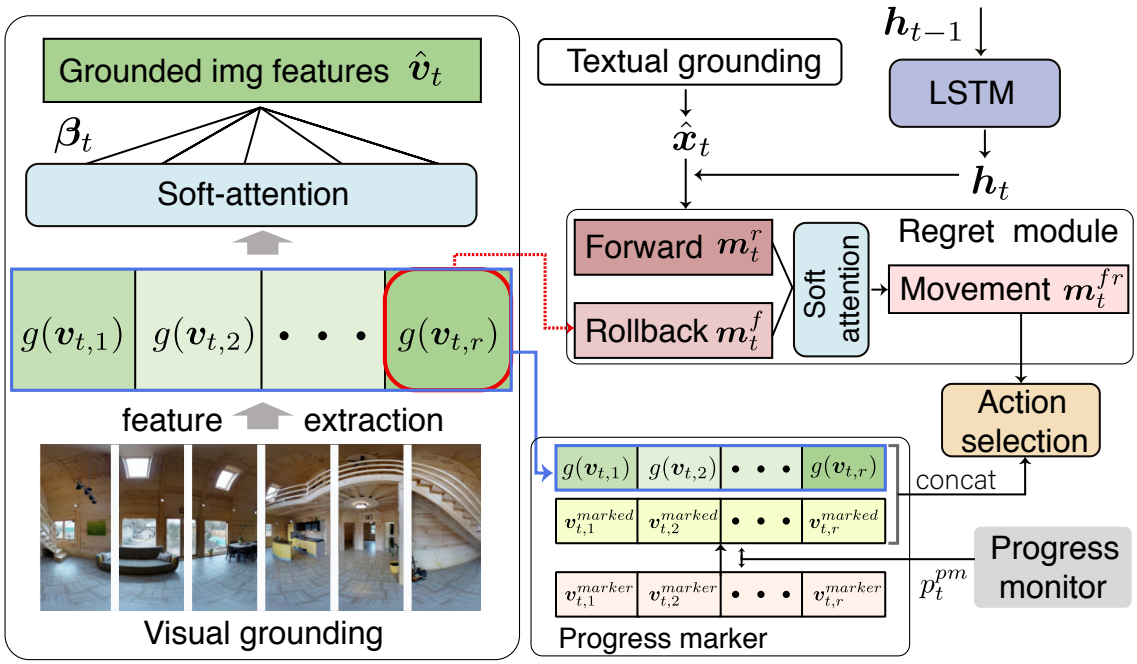

Figure 4.2: Illustration of the proposed regretful navigation agent. Note that the progress monitor is based on the self-monitoring agent in Chapter 3.

4.1.1 Regret Module

The **Regret Module** takes in the outputs of the progress monitor at different time steps and decides whether to go *forward* or to *rollback*. In particular, we use the concatenation of hidden state $\boldsymbol{h}_t$ and grounded instruction $\hat{\boldsymbol{x}}_t$ as our *forward* embedding $\boldsymbol{m}_t^f$, and more importantly we introduce a *rollback* embedding $\boldsymbol{m}_t^r$ to be the projection of the visual features for the action that leads to the previously visited location. The two vector representations are as follows:

$$\boldsymbol{m}_t^f = \boldsymbol{W}_a[\boldsymbol{h}_t, \hat{\boldsymbol{x}}_t] \quad \text{and} \quad \boldsymbol{m}_t^r = g(\boldsymbol{v}_{t,r}), \tag{4.1}$$

where $\boldsymbol{W}_a$ are the learned parameters, $\hat{\boldsymbol{x}}_t$ is the grounded instruction obtained from the textual grounding module, and $\boldsymbol{v}_{t,r}$ is the image feature vector representing a direction that points to the previously visited location.

To decide whether to go forward or rollback, the Regret Module leverages the difference of the progress monitor outputs between the current time step and the previous time step

$\Delta p_t^{pm} = p_t^{pm} - p_{t-1}^{pm}$. Intuitively, if the difference is larger than a certain threshold $\Delta p_t^{pm} > \sigma$, the agent should decide to take a *forward* action, and vice versa. Since it is hard to decide an optimal value for σ, we achieve this by computing attention weights $\boldsymbol{\alpha}_t^{fr}$ and perform a weighted sum on both *forward* and *rollback* embeddings. If the weight on *rollback* is larger, the agent is likely to be biased to take an action that leads to the last visited location. Formally, the weights can be computed as:

$$\boldsymbol{\alpha}_t^{fr} = \text{softmax}(\boldsymbol{W}_r(\Delta p_t^{pm})) \tag{4.2}$$

$$\boldsymbol{m}_t^{fr} = (\boldsymbol{\alpha}_t^{fr})^\top [\boldsymbol{m}_t^f, \boldsymbol{m}_t^r], \tag{4.3}$$

where $\boldsymbol{W}_r$ are the learnt parameters, $[,]$ denotes concatenation between feature vectors, and $\boldsymbol{m}_t^{fr}$ represents the weighted sum of the *forward* and *rollback* embeddings. Note that to ensure the progress monitor remains focused on estimating the agent's progress and regularizing the textual grounding module, we *detach* the output of the progress monitor which is fed into the Regret Module and set it as a leaf in the computational graph.

Action selection. Similar to existing work, the agent determines which image features from navigable directions have the highest correlation with the movement vector $\boldsymbol{m}_t^{fr}$ by computing the inner-product, and the probability of each navigable direction is then computed as:

$$\boldsymbol{o}_{t,k} = (\boldsymbol{W}_{fr}\boldsymbol{m}_t^{fr})^\top g(\boldsymbol{v}_{t,k}) \quad \text{and} \quad \boldsymbol{p}_t = \text{softmax}(\boldsymbol{o}_t), \tag{4.4}$$

where $\boldsymbol{W}_{fr}$ are the learned parameters and $\boldsymbol{p}_t$ is the probability distribution over navigable directions at time t. In practice, once the agent takes a rollback action, we block the action that leads to oscillation.

4.1.2 Progress Marker

The Regret Module provides a mechanism for the agent to decide when to rollback to a previous location or move forward according to the progress monitor outputs. Once the agent rolls back, it is required to select the next direction to go forward. It is thus essential for the agent to (1) know which directions it has already visited (and rolled back) and (2) estimate if the visited locations can lead to a path which completes the given instruction.

Toward this end, we propose the **Progress Marker** to mark each visited location with the agent's confidence in completing the instruction (see Figure 4.3). More specifically, we maintain a set of memory $\boldsymbol{M}$ and store the output of the progress monitor associated with each visited location; if the location is not yet visited, the marker will be filled with 1:

$$\boldsymbol{v}_{t,k}^{marker} = \begin{cases} p_i^{pm}, & \text{if } k \text{ leads to a location } i \in \boldsymbol{M}. \\ 1, & \text{otherwise.} \end{cases}$$

where i is a unique viewpoint ID for each location. We allow the marker on each location to be updated every time the agent visits it.

The marker value on each navigable direction indicates the estimated confidence that a location leads to the goal. We assign a value 1 for unvisited directions to encourage the agent to explore the environment. The navigating probabilities between unvisited directions depend on the action probabilities $\boldsymbol{p}_t$ since their marker values are the same.

Action selection with Progress Marker. During action selection, in addition to the movement vector $\boldsymbol{m}_t^{fr}$ that the agent can rely on in deciding which direction to go, we propose to label the marker value to each navigation direction as indications of whether a direction is likely to lead to the goal or to unexplored (and potentially better) paths. To achieve this, we leverage the difference between the current estimated progress and the marker for each navigable direction $\Delta \boldsymbol{v}_{t,k}^{marker} = p_t^{pm} - \boldsymbol{v}_{t,k}^{marker}$. We then concatenate it to

the visual feature representation for each navigable direction before action selection.

$$\boldsymbol{v}_{t,k}^{marked} = [g(\boldsymbol{v}_{t,k}), \Delta\boldsymbol{v}_{t,k}^{marker}]. \tag{4.5}$$

The difference $\Delta\boldsymbol{v}_{t,k}^{marker}$ indicates the chances of navigable directions leading to the goal and further inform the agent which direction to select. In our design, lower $\Delta\boldsymbol{v}_{t,k}^{marker}$ corresponds to higher chance for action selection. For instance, in step 4 in Figure 4.3, the $\Delta\boldsymbol{v}_{t,k}^{marker}$ for starting location and the last visited location are 0.08 and -0.02 respectively, whereas an unvisited location will have -0.71, which eventually leads to 0.52 estimated progress. When using Progress Marker, the final action selection is formulated as:

$$\boldsymbol{o}_{t,k} = (\boldsymbol{W}_{fr}\boldsymbol{m}_t^{fr})^\top \boldsymbol{v}_{t,k}^{marked} \quad \text{and} \quad \boldsymbol{p}_t = \text{softmax}(\boldsymbol{o}_t) \tag{4.6}$$

In practice, we tiled the difference n times before concatenating with the projected image feature $\boldsymbol{v}_{t,k}$ in order to account for imbalance. The marker value for the *stop* action is set to be 0.

4.1.3 Training and Inference

We train the proposed agent with cross-entropy loss for action selection and Mean Squared Error (MSE) loss for progress monitor. In addition to these losses, we also introduce an additional entropy loss to encourage the agent to explore other actions, such that it is not biased to actions with already very high confidence. The motivation is that, after training an agent for a period of time, the agent starts to overfit and perform fairly well on the training set. As a result, the agent will not learn to properly roll back during training since the majority of the training samples do not require the agent to roll back. Introducing the entropy loss increases the chance of exploration and making incorrect actions during

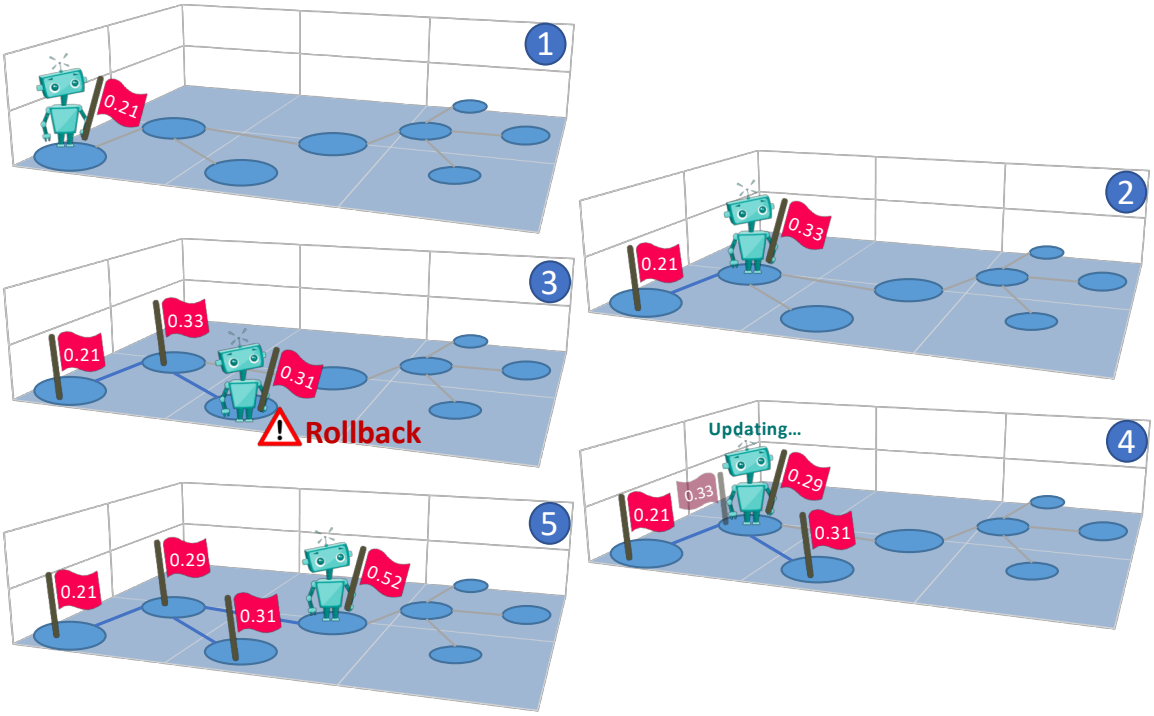

Figure 4.3: Concept of the proposed Progress Marker (red flags). The agent marks each visited location with estimated progress made towards the goal. The changes on the estimated progress determines whether the agent should *rollback* or *forward*, and the difference between the current estimated progress and the markers on the next navigable directions helps the agent decide which direction to go.

training.

$$\mathcal{L}_{loss} = \lambda \underbrace{\sum_{t=1}^{T} y_t^{nv} log(p_{t,k})}_{\text{action selection}} + (1-\lambda) \underbrace{\sum_{t=1}^{T} (y_t^{pm} - p_t^{pm})^2}_{\text{progress monitor}} - \beta \underbrace{\sum_{t=1}^{T} \sum_{k=1}^{K} -p_{t,k} log(p_{t,k})}_{\text{entropy loss}}, \quad (4.7)$$

where $p_{t,k}$ is the action probability of each navigable direction, y_t^{nv} is the ground-truth navigable direction at step t, $\lambda = 0.5$ is the weight balancing the cross-entropy loss and MSE loss, and $\beta = 0.01$ is the weight for entropy loss.

Following existing approaches [1, 2, 13], we perform categorical sampling during training for action selection. During inference, the agent greedily selects the action with highest action probability.

Table 4.1: Performance comparison with the state of the arts with greedy decoding for action selections. *: with data augmentation. Note that both Speaker-Follower [2] and Self-Monitoring (Chapter 3) were originally designed to optimize the success rate (SR) via beam search.

	Validation-Seen				Validation-Unseen				Test (unseen)			
Method	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Random	9.45	0.16	0.21	-	9.23	0.16	0.22	-	9.77	0.13	0.18	0.12
Student-forcing [13]	6.01	0.39	0.53	-	7.81	0.22	0.28	-	7.85	0.20	0.27	0.18
RPA [3]	5.56	0.43	0.53	-	7.65	0.25	0.32	-	7.53	0.25	0.33	0.23
Speaker-Follower [2]*	3.36	0.66	0.74	-	6.62	0.36	0.45	-	6.62	0.35	0.44	0.28
Self-Monitoring [1]*	**3.22**	0.67	**0.78**	0.58	5.52	0.45	0.56	0.32	5.99	0.43	0.55	0.32
Regretful	3.69	0.65	0.72	0.59	5.36	0.48	**0.61**	0.37	-	-	-	-
Regretful*	3.23	**0.69**	0.77	**0.63**	**5.32**	**0.50**	0.59	**0.41**	**5.69**	**0.48**	**0.56**	**0.40**

Table 4.2: Ablation study showing the effect of each proposed components compared to the prior arts. All methods here trained without data augmentation.

		Regret	Progress	Validation-Seen				Validation-Unseen			
Method	#	Module	Marker	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Speaker-Follower [2]				4.86	0.52	0.63	-	7.07	0.31	0.41	-
Self-Monitoring [1]				3.72	0.63	**0.75**	0.56	5.98	0.44	0.58	0.30
Regretful	1	✓		3.88	0.64	0.70	0.58	5.65	0.47	0.59	**0.37**
	2		✓	3.76	0.63	0.73	0.57	5.74	0.44	0.59	0.32
	3	✓	✓	**3.69**	**0.65**	0.72	**0.59**	**5.36**	**0.48**	**0.61**	**0.37**

4.2 Experimental Results

4.2.1 Comparison with Prior Art.

We first compare the proposed regretful navigation agent with the state-of-the-art methods. As shown in Table 4.1, our method achieves significant performance improvement over the existing approaches. We achieved 37% SPL and 48% SR on the validation unseen set and outperformed all existing work. Our best performing model achieves 41% SPL and 50% SR on validation unseen set when trained with the synthetic data generated from the Speaker [2]. We demonstrate absolute 8% SPL improvement and 5% SR improvement on the test server over the current state-of-the-art method. We can also see that our regretful navigation agent without data augmentation has already outperformed the existing work on both SR and SPL metrics.

Table 4.3: Sanity check for verifying that the source of performance improvement is from the agent's ability to decide when to roll back.

Method	Blocking Rollback	Validation-Seen				Validation-Unseen			
		NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Self-Monitoring [1]		3.72	0.63	**0.75**	0.56	5.98	0.44	0.58	0.30
	✓	3.85	0.64	**0.75**	0.58	6.02	0.44	0.60	0.34
Regretful		**3.69**	**0.65**	0.72	0.59	**5.36**	**0.48**	**0.61**	0.37
	✓	3.91	0.64	0.68	**0.60**	5.80	0.46	0.55	**0.41**

4.2.2 Ablation Study

Table 4.2 shows an ablation study to analyze the effect of each component. The first thing to note is that our method is significantly better than the Self-Monitoring agent which uses greedy decoding, even though it still has a progress monitor loss (although the progress monitor is not used for action selection). A second interesting point is that when the Progress Marker is available with the features of each navigable direction that have been visited before, but the Regret Module is not available, performance does not increase significantly (44% SR). Note that we also conducted an experiment with another condition, where the progress monitor estimates were attached to the *forward* embedding, meaning that the network could use that information to improve action selection. That condition again was only able to achieve modest gains (45% SR), compared to our Regret Module which was able to achieve 47% SR (and 48% when the Progress Marker was added). In all, this shows that the key improvement stems from the design of the Regret Module, allowing the agent to intelligently backtrack after making mistakes.

Does rollback lead to the performance improvement? Our proposed regretful agent relies on the ability to regret and roll back to a previous location, further exploring the unknown environment to increase the success rate. As a sanity check, we manually block all actions leading to rollback for both the state-of-the-art Self-Monitoring agent and our regretful agent[1]. The result is shown in Table 4.3. As can be seen, blocking rollback for the Self-Monitoring agent produces mixed results, with worse NE but better metrics such

[1] except when there is only one navigable direction to go.

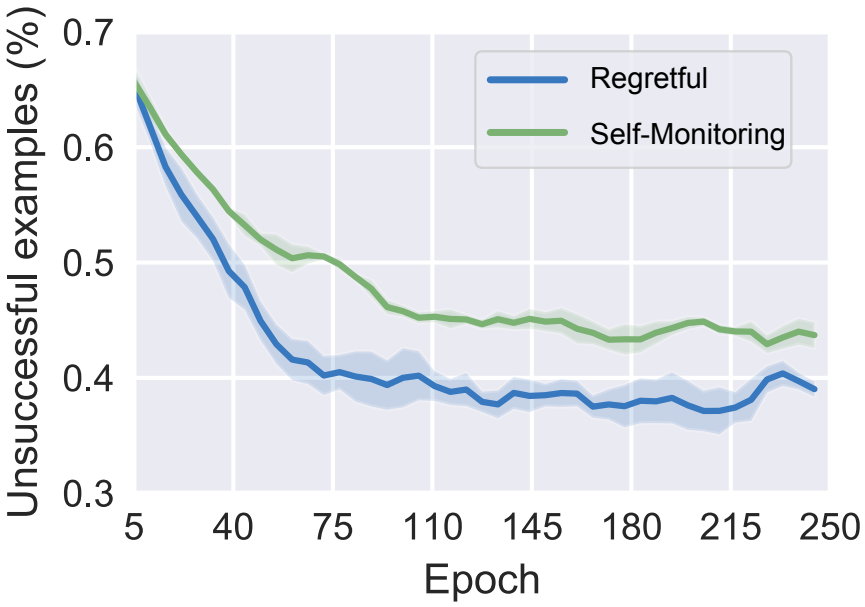

Figure 4.4: Percentage of unsuccessful examples involving rollback reduced by our proposed regretful agent.

as OSR. The SR, however, is unchanged. On the other hand, blocking rollback for our agent significantly reduces most metrics including NE, SR, and OSR especially on unseen environments. This shows that blocking the ability to learn when to roll back degrades a large source of performance increase, and this is especially true for unseen environments.

Number of unsuccessful examples reduced. We calculate the total number of unsuccessful examples involves rollback action for both Self-Monitoring and our proposed agent (in percentage). As demonstrated in Figure 4.4, our proposed regretful agent significantly reduces the unsuccessful examples from around 43% to 38%, which correlates to the 4-5% improvement on SR in Table 3.1 and 4.2.

Regretful agent in unfamiliar environments. The key to the performance increase of an agent focusing on the rollback ability is not that the agent learns a better textual or visual grounding, but that the agent learns to search especially when it is not certain which direction to go. To demonstrate this, we train both the Self-Monitoring agent and our proposed regretful agent only on synthetic data and test them on the unseen validation set (real data). We expect the regretful agent to outperformed the Self-Monitoring agent across

Table 4.4: Ablation study when trained using only the synthetic or real training data. Oracle Navigation Error (ONE): the navigation error if the agent can stop at the closest point to the goal along its trajectory.

			Validation-Unseen		
Method	Synthetic	Real	ONE ↓	SR ↑	OSR ↑
Self-Monitoring	✓		4.09	0.35	0.49
		✓	3.62	0.44	0.58
Regretful	✓		3.47	0.41	0.58
		✓	**2.26**	**0.48**	**0.61**

all metrics since our agent is designed to operate in an environment where the agent is likely to be uncertain on action selection. As shown in Table 4.4, when trained using only the synthetic data, our method significantly outperformed Self-Monitoring agent. Interestingly, when compared with the Self-Monitoring agent trained with real data, our agent trained with synthetic data is slightly better on ONE, same on OSR, and marginally lower on SR. We achieved slightly better performance on oracle metrics since stopping at the correct location is not a hard constrain. This indicates that even though our regretful agent is not yet learned how to properly stop at the goal (due to training on synthetic data only), the chance that it passes/reaches the goal is slightly higher than Self-Monitoring agent trained with real data. Further, when the regretful agent trained with real data, the performance improved across all metrics.

4.3 Qualitative Results

4.3.1 Successful examples

We show the complete trajectory of the agents successfully deciding when to roll back and reach the goal in unseen environments in Figure 4.5, 4.6, 4.7, and 4.8.

In Figure 4.5, we demonstrate that the agent is capable of performing a local search on the navigation graph. Specifically, from step 0 to step 3, the agent searched two possible directions and decided to move with one particular direction at step 4. Once it reached

step 5, the agent decides to continue to move forward, and we observed that the progress estimate significantly increased to 45% at step 7. Interestingly, unlike other examples we have shown, the agent did not decide to roll back despite the progress estimate slightly decreased from 45% to 40%. We reckon that this is one of the advantages of using a learning-based regret module, where a learned and dynamically changing threshold decides when to rollback. Finally, the agent successfully stopped in front of the microwave.

In Figure 4.6, the agent is instructed to *walk across living room*. It is ambiguous since both directions seem like a living room. Our agent first decides to move into the direction that leads to a room with a kitchen and living room. It then decided to roll back with the progress monitor output slightly decreased. The agent then followed the rest of the instruction successfully with the progress monitor steadily increased at each step after that. Finally, the agent decides to stop with the progress estimate 99%.

In Figure 4.7, the agent first moved out of the room and walked up the stairs as instructed, but the second set of stairs makes the instruction ambiguous. The agent continued to walk up the stairs for one more step and then decided to go down the stairs at step 4. As the agent decided to turn right at step 6, we can see the progress estimate significantly increased from 51% to 66%. Once the agent entered the TV room, the progress estimate increased again to 82%. Finally, the agent successfully stopped with the progress monitor output 95%.

In Figure 4.8, the agent failed to *walk down the stairs* at step 1. Because of the proposed Regret Module and Progress Marker, the agent was able to discover the correct path to go downstairs. Once walking down, the progress estimate increased to 39% immediately, and as the agent goes further down, the progress estimate reached 98% by the time the agent reached the bottom of the stairs. Finally, the agent decided to wait by the bamboo plant with progress estimate 99%.

4.3.2 Failed examples

We have shown how the agent can successfully utilize the rollback mechanism to reach the goal, even though it is not familiar with the environment and likely to be uncertain about some actions it took. Intuitively, the rollback mechanism can increase the chance that the agent reaches the goal as long as the agent can correctly decide when to stop.

We now discuss two failed examples of our proposed regretful agent in unseen environments that highly resemble the successful examples in terms of the given instruction and ground-truth path. Both examples demonstrate that the agent successfully rolled back to the correct path towards the goal but failed to stop at the goal.

Specifically, in Figure 4.9, the agent reaches the room with the white cabinet as instructed but decided to move one step forward. The agent then decided to roll back to the room correctly at step 5. However, this does not help the agent to stop at the goal resulting in a failed run.

On the other hand, in Figure 4.10, we can see that the progress estimate at step 5 significantly dropped by 21%, and the agent correctly decided to roll back. The agent then successfully reached the refrigerator but did not stop immediately. It continued to move forward after step 8, resulting in an unsuccessful run.

Lastly, we discuss a failed example when the agent incorrectly decided when to roll back. In Figure 4.11, the agent first followed the instruction to *go down the hallway* and tried to find the second door to turn right. As the agent reached the end of the hallway at step 4, it decided to roll back since there is no available navigable direction that leads to *turn right*. The agent then decided to go down the hallway again with completely opposite direction. However, the agent decided to roll back again at step 7 with the progress estimate dropped to 18%. Although the agent eventually was able to *escape* from the hallway leading to the dead end, it ends up unsuccessful.

4.4 Summary

In this chapter, we have proposed an end-to-end trainable regretful navigation agent for the VLN task. Inspired by the intuition of viewing this task as graph search over the navigation graph, we use a progress monitor as a learned heuristic that can be trained and employed during inference to greedily select the next best action (best-first search). We then propose a Regret Module that is able to learn to decide when to perform backtracking depending on the progress made and state of the agent. Finally, a Progress Marker is used to allow the agent to reason about previous visits and unvisited directions, so that the agent can choose a better navigable direction by reducing action probabilities for visited locations with lower progress estimate. The resulting framework achieved state-of-the-art SR and SPL compared to existing methods without using beam search on the public leaderboard.

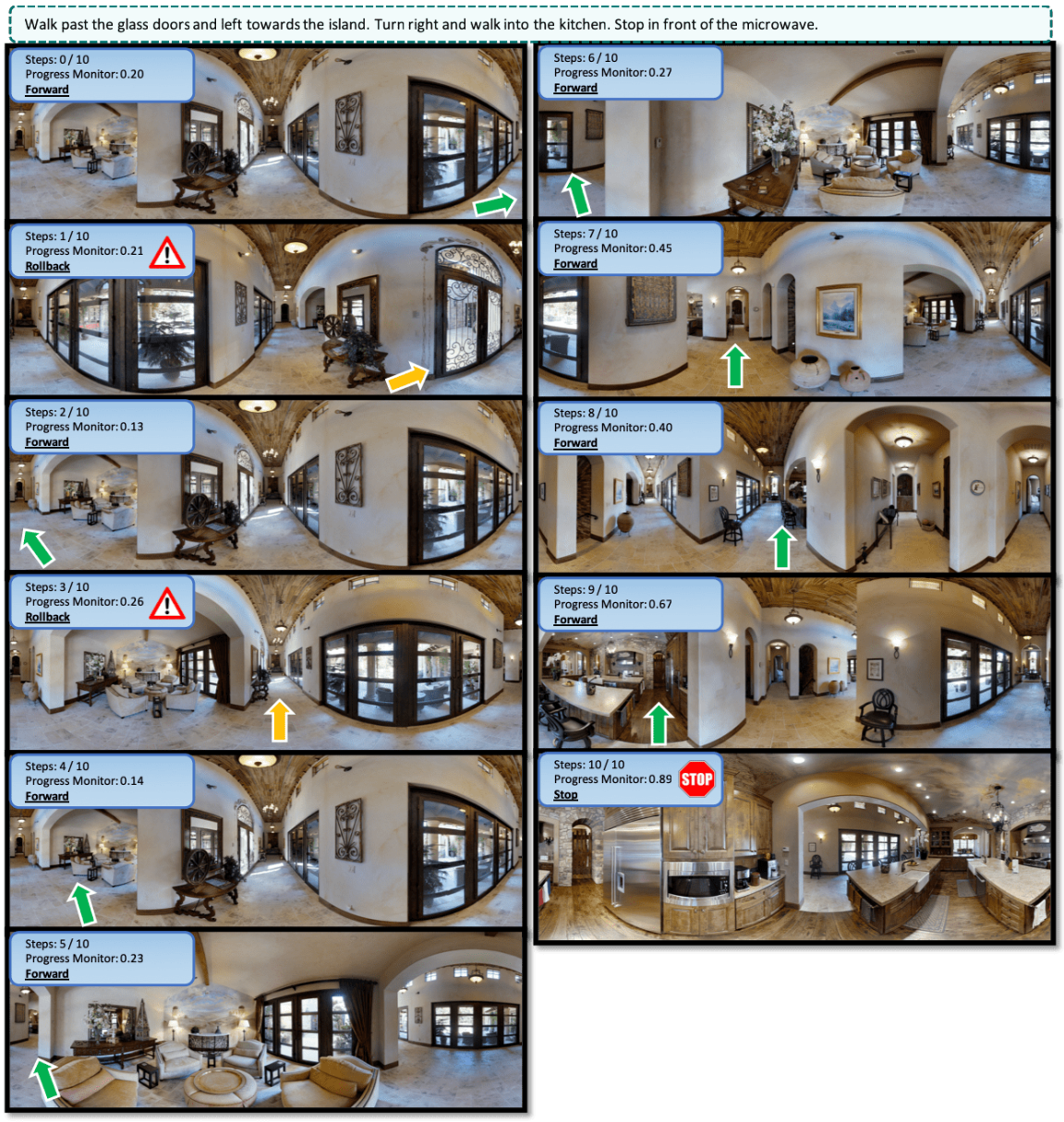

Figure 4.5: The first part of the instruction *walk past the glass doors* is ambiguous since there are multiple directions that lead to glass doors, and naturally the agent is confused and uncertain where to go. Our agent is able to perform local search on the navigation graph and decides to roll back multiple times at the beginning of the navigation. At step 6, the agent performs an action *turn right*. Consequently, the progress estimate at step 7 significantly increased to 45%. Interestingly, the agent continues to move forward even though the progress estimate slightly decreased from step 7 to step 8. We reckon that this as one of the advantage of using a learning-based regret module as opposed to using a hard-coded threshold. The agent then successfully follows the instruction and *stops in front of the microwave* with progress estimate 89%.

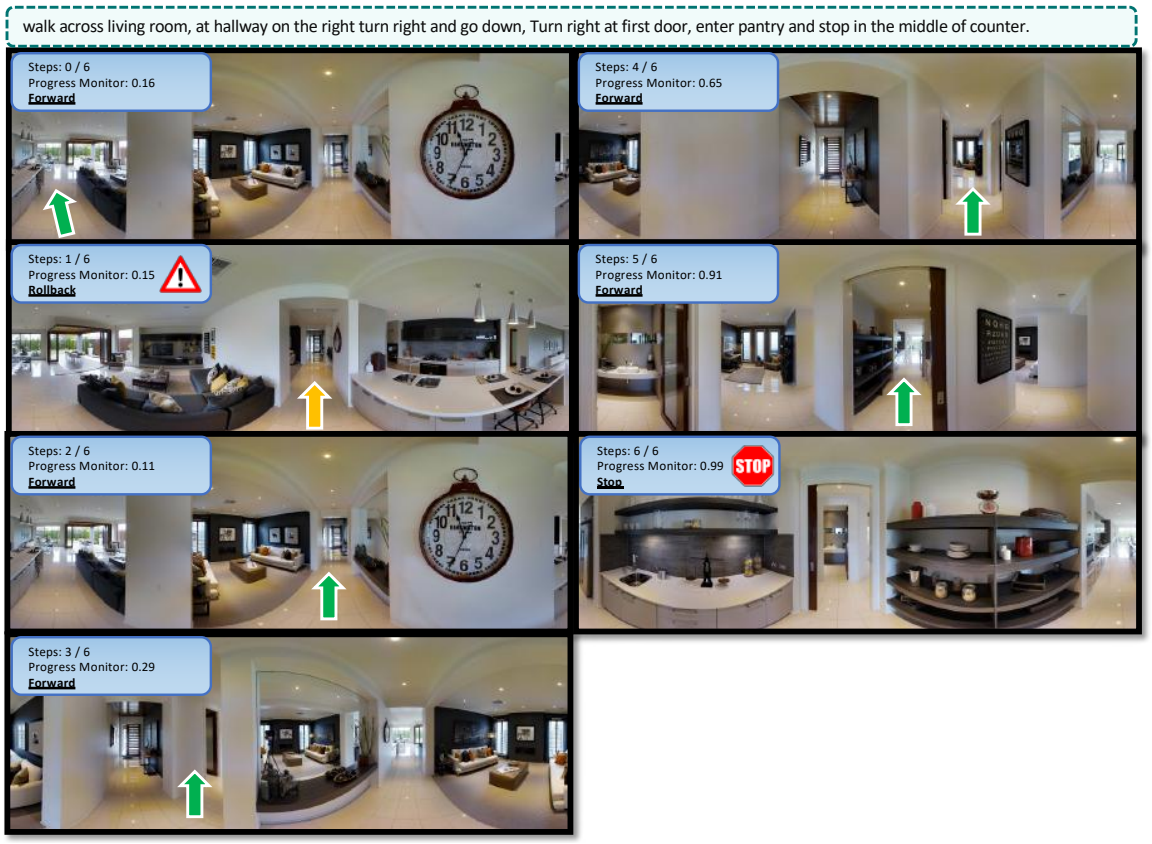

Figure 4.6: The agent first *walk across living room*, but decides to move into the direction that leads to kitchen and dinning room. At step 1, the agent decides to roll back due to a decreasing of the progress monitor output. The agent then followed the rest of the instruction successfully with the progress monitor steadily increased at each step. Finally, the agent decides to stop with the progress estimate 99%.

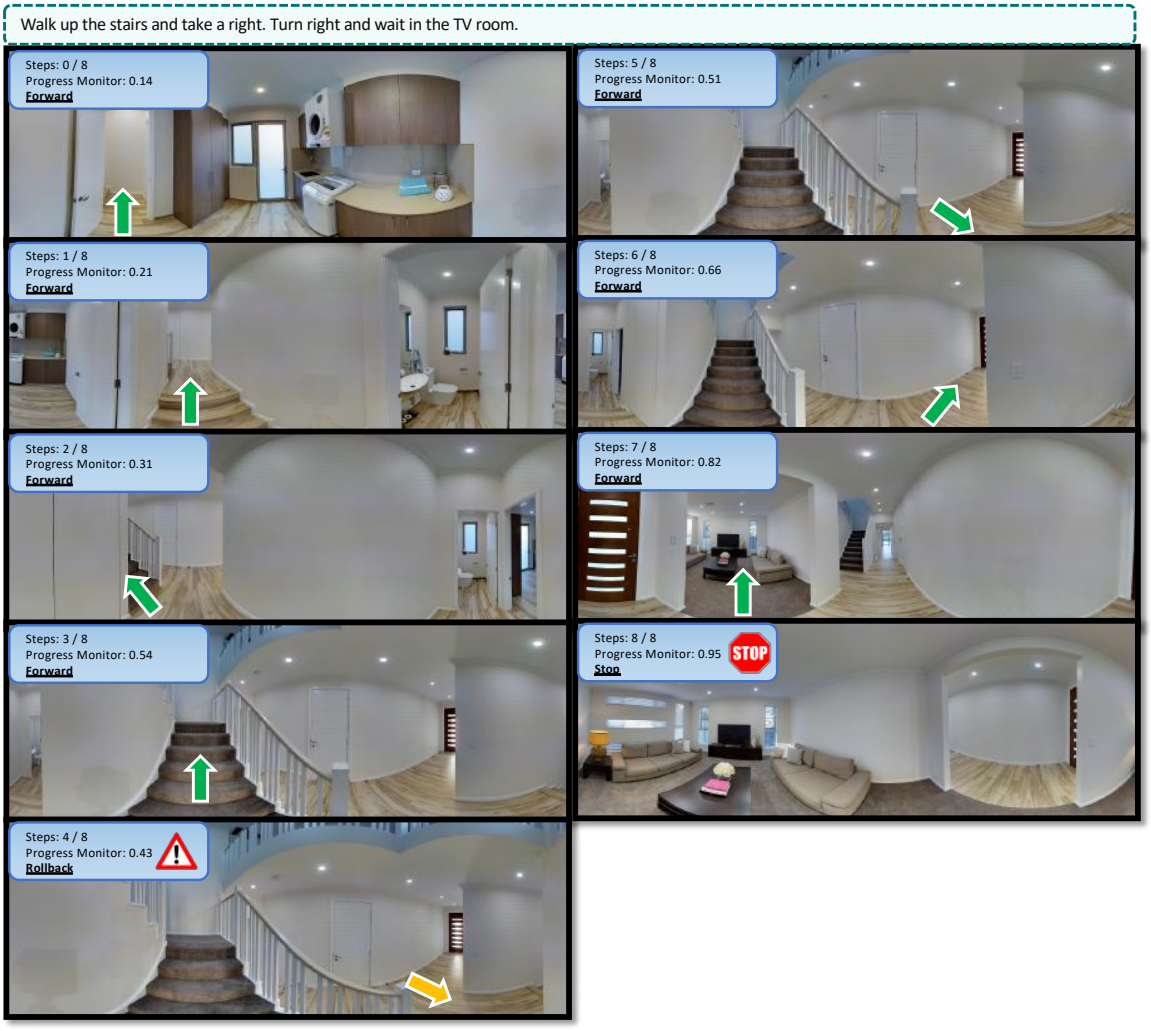

Figure 4.7: The agent walked up the stairs as instructed at step 1, but the second set of stairs makes the instruction ambiguous. The agent continues to walk up stairs but soon realized that it needs to go down the stairs and *turn right* from step 4 - 6. When the agent decides to turn right, we can see the progress estimate significantly increased from 51% to 66%. As the agent turned right to the TV room, the progress estimate increased again to 82%. Finally, the agent stops with the progress monitor output 95%.

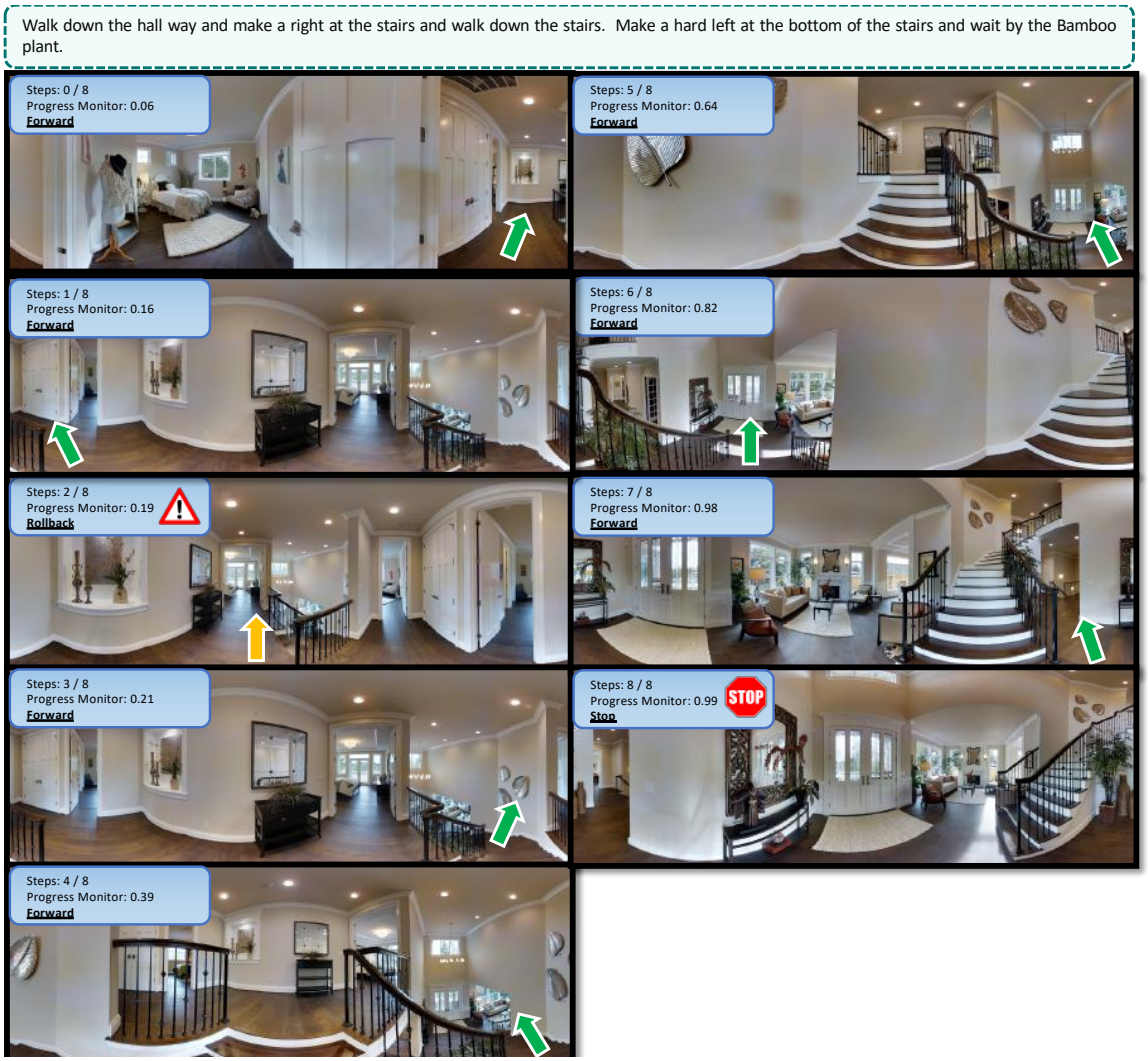

Figure 4.8: The agent walks down the hall way to the stairs but failed to *walk down the stairs* at step 1. With a small increase on the progress monitor output, the agent then decides to roll back and take the action to walk down the stairs. Once walking down, we can see the progress estimate increased to 39%, and as the agent goes further down, the progress estimate reached 98% at the bottom of the stairs. Finally, the agent decides to stop near by the bamboo plant with progress estimate 99%.

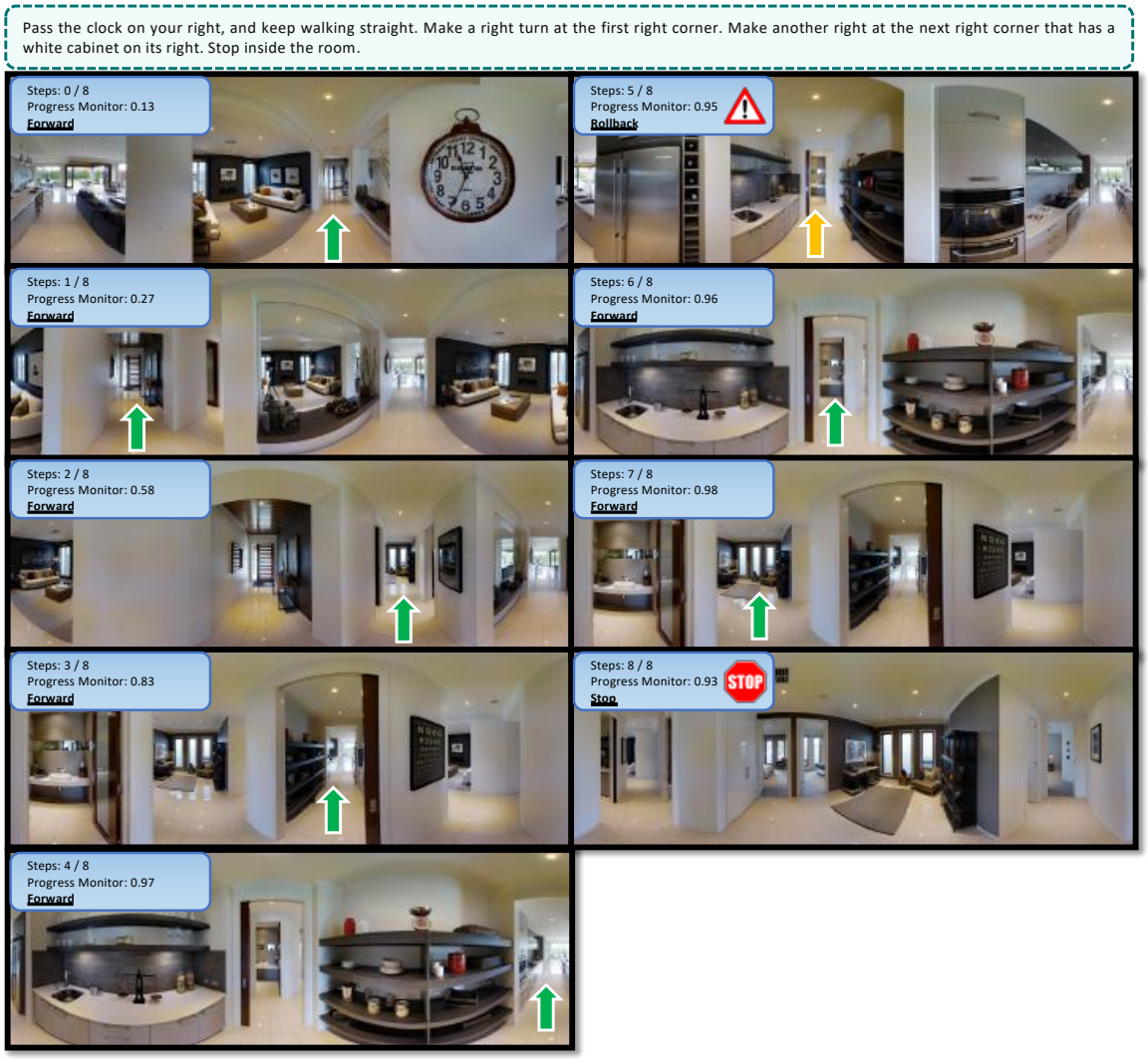

Figure 4.9: **Failed example.** The agent starts to navigate through the unseen environment by following the given instruction. It was able to successfully follow the instruction and correctly reach the goal at step 4. The agent then decided to move forward towards the kitchen and correctly decided to roll back to the goal. However, the agent did not stop and continue to explore the environment and eventually stopped a bit further from the goal.

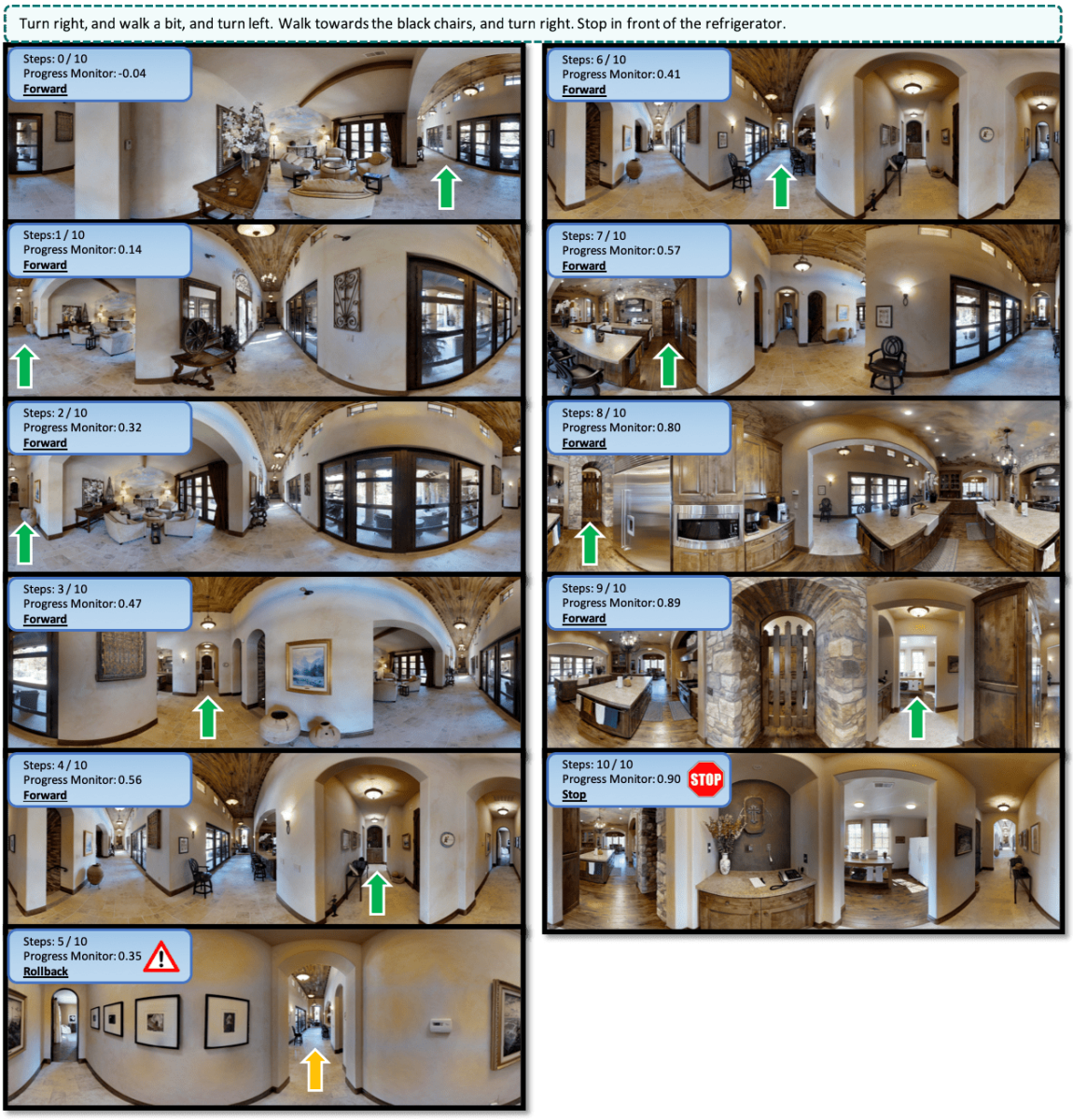

Figure 4.10: The agent correctly followed the first parts of the instruction until step 4, but it decided to move forward towards the hall. At step 5, the agent correctly decided to roll back with the progress estimate decreased from 56% to 35%. The agent was then able to follow the rest of the instruction successfully and reach the refrigerator at step 8. However, the agent did not stop nearby the refrigerator and continued to take another two forward steps.

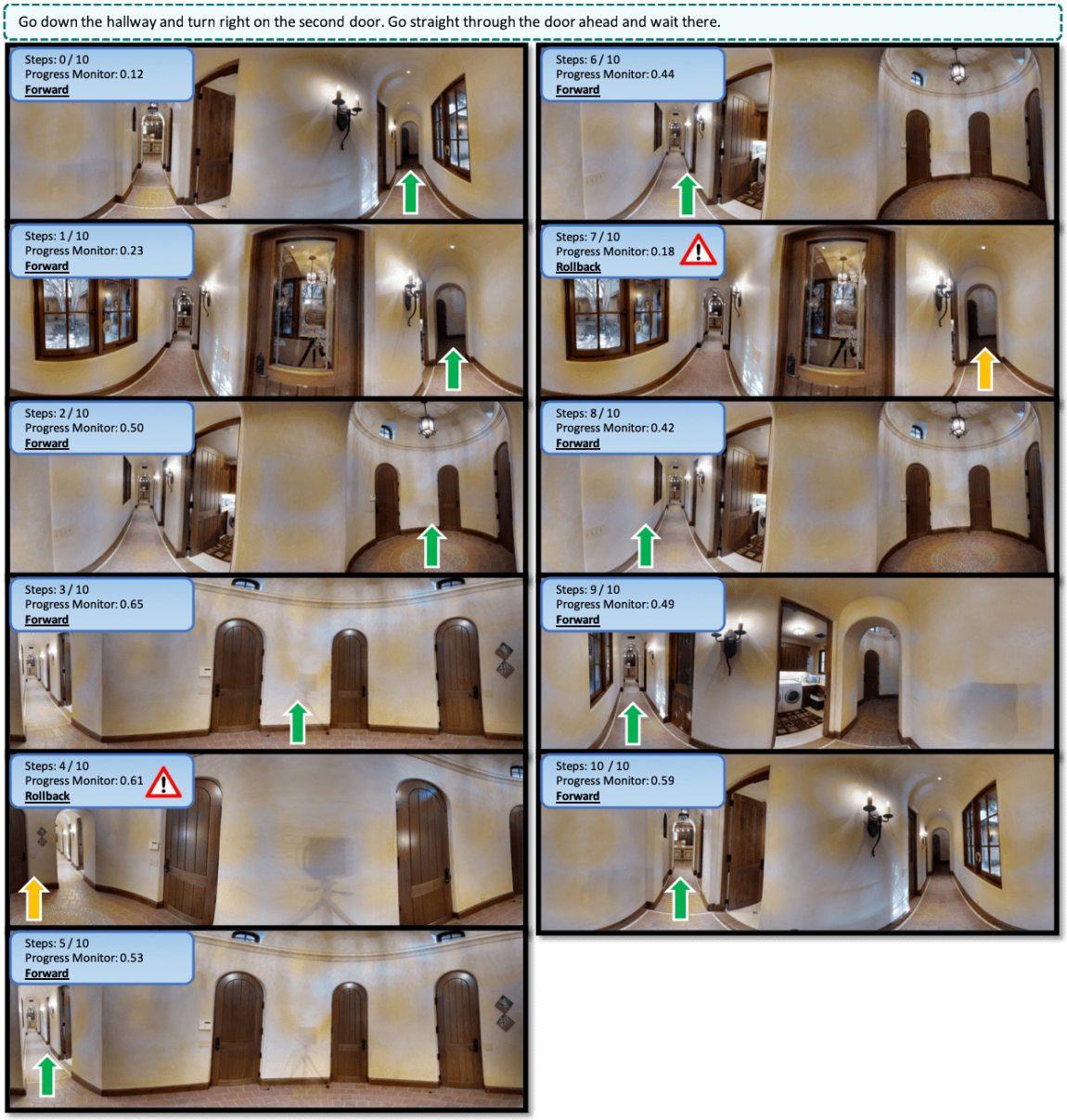

Figure 4.11: The agent followed the first part of instruction to *go down the hallway*. As the agent reached the end of the hallway, it was not able to find the second door to turn left. The agent then decided to roll back at step 4 with progress estimate decreased from 65% to 61%. The agent continued to go back towards the hallway but decided to roll back again at step 7. Although the agent was able to correct its errors made at the first few steps and *escape* from the hallway leading to the dead end, it ends up unsuccessful.

CHAPTER 5

OBJECT-LEVEL FINE-GRAINED VIDEO UNDERSTANDING

Video understanding tasks such as activity recognition and caption generation are crucial for various applications in surveillance, video retrieval, human behavior understanding, etc. Recently, datasets for video understanding such as Charades [131], Kinetics [132], and ActivityNet Captions [133] contain diverse real-world examples and represent complex human and object interactions that can be difficult to model with state-of-the-art video understanding methods [131]. Consider the example in Figure 5.1. To accurately predict *cooking on campfire* and *cooking egg* among other similar action classes requires understanding of fine-grained object relationships and interactions. For example, a hand breaks an egg, eggs are in a bowl, the bowl is on top of the campfire, campfire is a fire built with wood at a camp, etc. Although recent state-of-the-art approaches for action recognition have demonstrated significant improvements over datasets such as UCF101 [134], HMDB51 [135], Sports-1M [136], THUMOS [137], ActivityNet [138], and YouTube-8M [139], they often focus on representing the overall visual scene (coarse-grained) as sequence of inputs that are combined with temporal pooling, e.g. CRF, LSTM, 1D Convolution, attention, and NetVLAD [14, 140, 15, 16], or use 3D Convolution for the whole video sequence [17, 18, 19]. These approaches ignore the fine-grained details of the scene and do not infer interactions between various objects in the video. On the other hand, in video captioning tasks, although prior approaches use spatial or temporal attention to selectively attend to fine-grained visual content in both space and time, they too do not model object interactions.

Prior work in understanding visual relationships in the image domain has recently emerged as a prominent research problem, *e.g.,* scene graph generation [76, 77] and visual relationship detection [72, 78, 73, 79, 81, 82]. However, it is unclear how these techniques can be adapted to open-domain video tasks, given that the video is intrinsically more com-

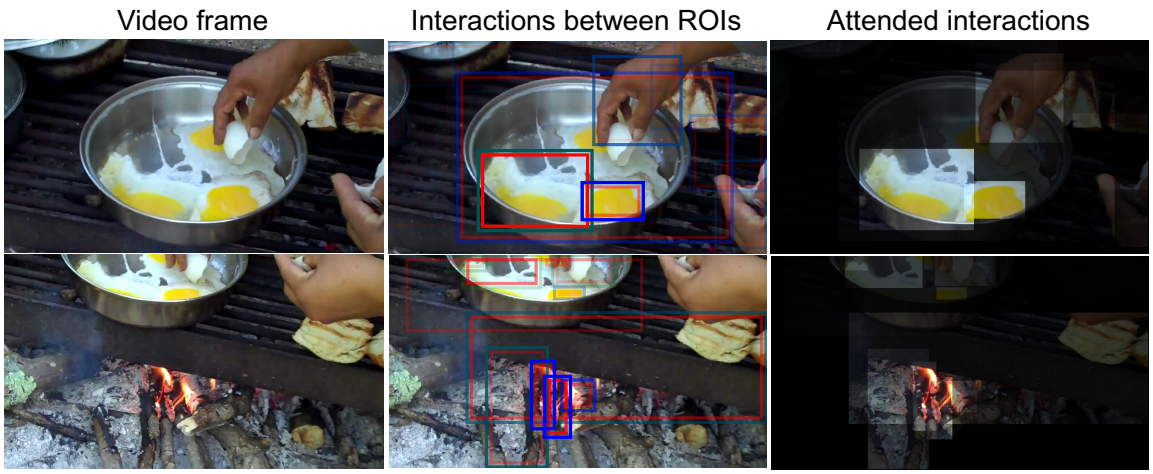

Figure 5.1: *Higher-order object interactions* are progressively detected based on selected inter-relationships. ROIs with the same color (weighted r, g, b) indicating there exist inter-object relationships, *e.g.,* eggs in the same bowl, hand breaks egg, and bowl on top of campfire (interaction within the same color). Groups of inter-relationships then jointly model higher-order object interaction of the scene (interaction between different colors). *Right*: ROIs are highlighted with their attention weights for higher-order interactions. The model further reasons about the interactions through time and predicts *cooking on campfire* and *cooking egg*. Images are generated from SINet (best viewed in color).

plicated in terms of temporal reasoning and computational demands. More importantly, a video may consist of a large number of objects over time. Prior approaches on visual relationship detection typically model the full pairwise (or triplet) relationships. While this may be realized for images, videos often contain hundreds or thousands of frames. Learning relationships across multiple objects alongside the temporal information is computationally infeasible on modern GPUs, and performance may suffer due to the fact that a finite-capacity neural network is used to model a large combinatorial space. Furthermore, prior work in both image and video domains [83, 84] often focus on pairwise relationships or interactions, where interactions over groups of interrelated objects—*higher-order interactions*—are not explored, as shown in Figure 5.2.

Toward this end, we present a generic recurrent module for fine-grained video understanding, which dynamically discovers higher-order object interactions via an efficient dot-product attention mechanism combined with temporal reasoning. Our work is applicable to

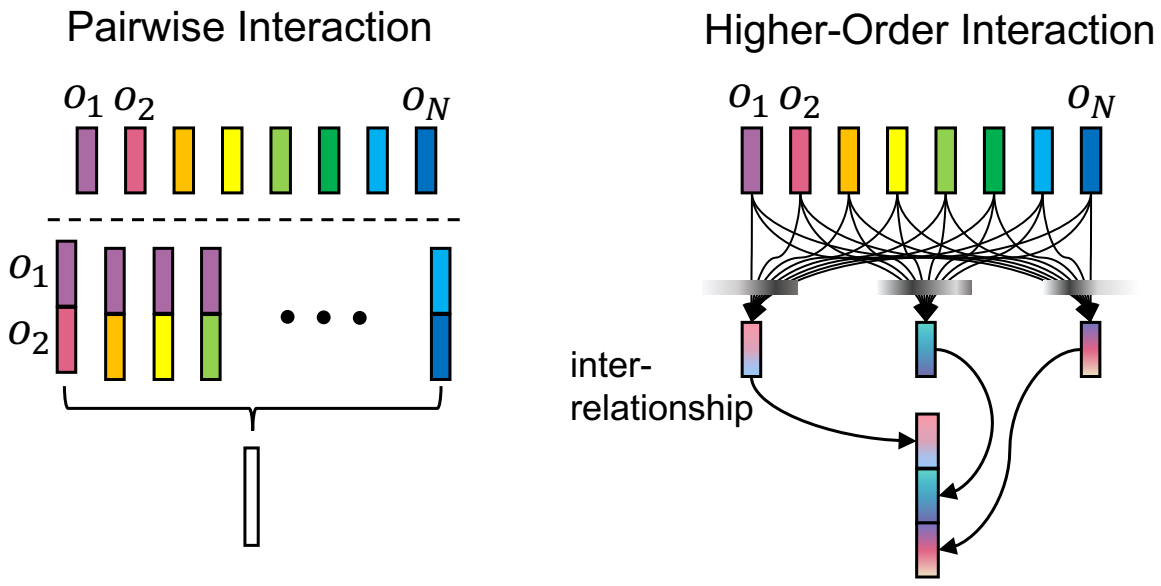

Figure 5.2: Typically, object interaction methods focus on pairwise interactions (left). We efficiently model the *higher-order interactions* between arbitrary subgroups of objects for video understanding, in which the inter-object relationships in one group are detected and objects with significant relationships (i.e. those that serve to improve action recognition or captioning in the end) are attentively selected (right). The higher-order interaction between groups of selected object relationships are then modeled after concatenation.

various open domain video understanding problems. In this paper, we validate our method on two video understanding tasks with new challenging datasets: action recognition on Kinetics [132] and video captioning on ActivityNet Captions [133] (with ground truth temporal proposals). By combining both coarse- and fine-grained information, our **SINet** (Spatiotemporal Interaction Network) for action recognition and **SINet-Caption** for video captioning achieve state-of-the-art performance on both tasks while using RGB video frames sampled at only maximum 1 FPS. To the best of our knowledge, this is the first work of modeling object interactions on open domain large-scale video datasets, and we also show that modeling higher-order object interactions can further improve the performance at low computational costs.

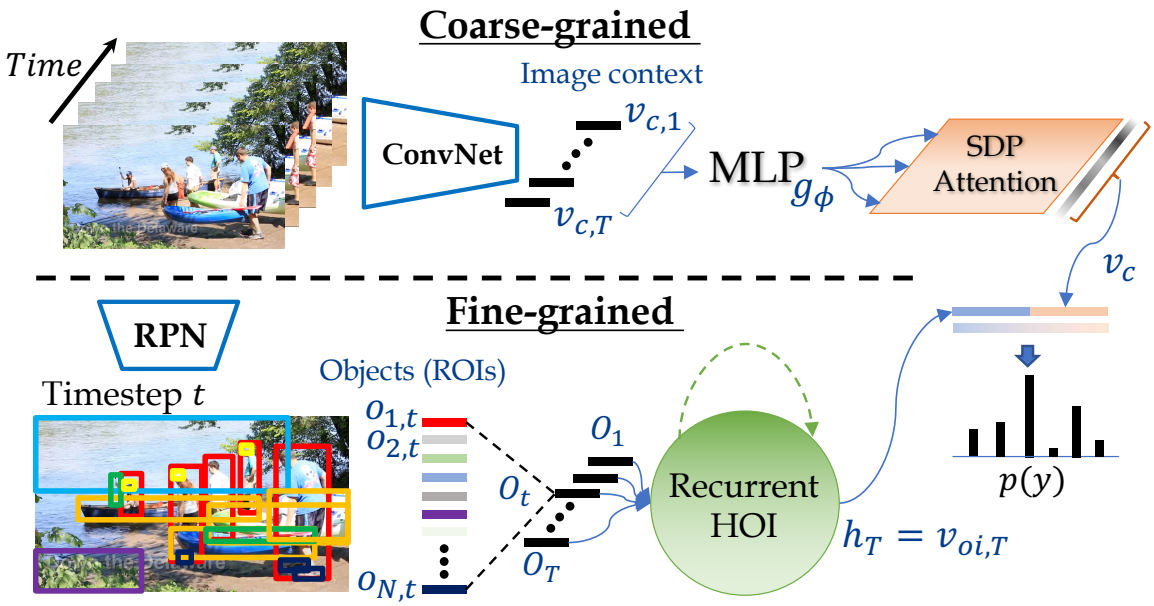

Figure 5.3: Overview of the SINet for action recognition. **Coarse-grained:** each video frame is encoded into a feature vector $v_{c,t}$. The sequence of vectors are then pooled via temporal SDP-Attention into single vector representation v_c. **Fine-grained:** Each object (ROI) obtained from RPN is encoded in a feature vector $o_{n,t}$. We detect the higher-order object interaction using the proposed generic recurrent Higher-Order Interaction (HOI) module. Finally, coarse-grained (image context) and fine-grained (higher-order object interactions) information are combined to perform action prediction.

5.1 Fine-grained Action Recognition

5.1.1 Coarse-grained image context

As recent studies have shown, using LSTM to aggregate a sequence of image representations often results in limited performance since image representations can be similar to each other and thus lack temporal variances [139, 132, 4]. As shown in Figure 5.3 (top), we thus begin by attending to key image-level representations to summarize the whole video sequence via the Scale Dot-Product Attention (SDP-Attention) [126]:

$$\alpha_c = softmax(\frac{{X_c}^\top X_c}{\sqrt{d_\phi}}), \quad X_c = g_\phi(V_c) \tag{5.1}$$

$$v_c = \overline{\alpha_c \, {X_c}^\top} \tag{5.2}$$

where V_c is a set of image features: $V_c = \left\{v_{c,1}, v_{c,2}, ..., v_{c,T}\right\}, v_{c,t} \in \mathbb{R}^m$ is the image feature representation encoded via a ConvNet at time t, and t ranges from $\left\{1, 2, ..., T\right\}$ for a given video length. g_ϕ is a Multi-Layer Perceptron (MLP) with parameter ϕ, d_ϕ is the dimension of last fully-connected (FC) layer of g_ϕ, $X_c \in \mathbb{R}^{d_\phi \times T}$ is the projected image feature matrix, $\sqrt{d_\phi}$ is a scaling factor, and $\alpha_c \in \mathbb{R}^{T \times T}$ is an attention weight applied to the (projected) sequence of image representations V_c. The weighted image representations are then mean-pooled to form video representation v_c.

5.1.2 Fine-grained higher-order object interactions

Traditional pairwise object interactions only consider how each object interacts with another object. We instead model inter-relationships between arbitrary subgroups of objects, the members of which are determined by a learned attention mechanism, as illustrated in Figure 5.2. Note that this covers pair-wise or triplet object relationships as a special case, in which the learned attention only focus on one single object. We define *objects* to be a certain region in the scene that might be used to determine the visual relationships and interactions.

Problem Statement. We define *objects* to be a certain region in the scene that might be used to determine the visual relationships and interactions. Each object representation can be directly obtained from an RPN and further encoded into an object feature. Note that we do not encode object class information from the detector into the feature representation since there exists a cross-domain problem, and we may miss some objects that are not detected by the pre-trained object detector. Also, we do not know the corresponding objects across time since linking objects through time can be computationally expensive for long videos. As a result, we have variable-lengths of object sets residing in a high-dimensional space that spans across time. Our objective is to efficiently detect higher-order interactions from these rich yet unordered object representation sets across time.

In the simplest setting, an interaction between objects in the scene can be represented

via summation operation of individual object information. For example, one method is to add the learnable representations and project these representations into a high-dimensional space where the object interactions can be exploited by simply summing up the object representations. Existing approaches of modeling relationships which have been widely used with images is by pairing all possible object candidates (or subject-object pairs) [72, 78, 79, 80, 81]. However, this is infeasible for video, since a video typically contains hundreds or thousands of frame and the set of object-object pairs is too large to fully represent. Detecting object relationships frame by frame is computationally expensive, and the temporal reasoning of object interactions is not used.

5.1.3 Recurrent Higher-Order Interaction Module

To overcome these issues, we propose a generic recurrent module for detecting higher-order object interactions for fine-grained video understanding problems, as shown in Figure 5.4. The proposed recurrent module dynamically selects object candidates which are important to discriminate the human actions. The combinations of these objects are then concatenated to model higher order interaction using group to group or triplet groups of objects.

First, we introduce learnable parameters for the incoming object features via MLP projection g_{θ_k}, since the object features are pre-trained from another domain and may not necessarily present interactions towards action recognition. The projected object features are then combined with overall image content and previous object interaction to generate K sets of weights to select K groups of objects[1]. Objects with inter-relationships are selected from an attention weight, which generates a probability distribution over all object candidates. The attention is computed using inputs from current (projected) object features, overall image visual representation, and previously discovered object interactions

[1]The number K depends on the complexity of the visual scene and the requirement of the task (in this case, action recognition). We leave dynamically selecting K to future work.

(see Figure 5.4), which provide the attention mechanism with maximum context.

$$\alpha_k = Attention(g_{\theta_k}(O_t), v_{c,t}, h_{t-1}) \tag{5.3}$$

where the input O_t is a set of *objects*: $O_t = \{o_{1,t}, o_{2,t}, ..., o_{N,t}\}, o_{n,t} \in \mathbb{R}^m$ is the n^{th} object feature representation at time t. The g_{θ_k} is a MLP with parameter θ_k, the parameters are learnable synaptic weights shared across all objects $o_{n,t}$ and through time t. $v_{c,t}$ denotes as encoded image feature at current time t, and h_{t-1} is the previous output of LSTM cell which represents the previous discovered object interaction. Formally, given an input sequence, a LSTM network computes the hidden vector sequences $\mathbf{h} = (h_1, h_2, ..., h_T)$. Lastly, α_k is an attention weight computed from the proposed attention module.

Attentive selection module. Here we discuss two possible choices for the attention module, as shown in Figure 5.5. Dot-product attention considers inter-relationships when selecting the objects, and α-attention does not.

- Dot-product attention. In order to model higher-order interactions, which models inter-object relationships in each group of selected objects, we use dot-product attention since the attention weights computed for each object is the combination of all objects.

Formally, the current image representation $v_{c,t}$ and the last object interaction representation h_{t-1} are first projected to introduce learnable weights. The projected $v_{c,t}$ and h_{t-1} are then repeated and expanded N times (the number of objects in O_t). We directly combine this information with projected objects via matrix addition and use it as input to dot-product attention. We added a scale factor as in [126]. The input to the first matrix multiplication and the attention weights over all objects can be defined as:

$$X_k = repeat(W_{h_k} h_{t-1} + W_{c_k} v_{c,t}) + g_{\theta_k}(O_t) \tag{5.4}$$

$$\alpha_k = softmax(\frac{{X_k}^\top X_k}{\sqrt{d_\theta}}) \tag{5.5}$$

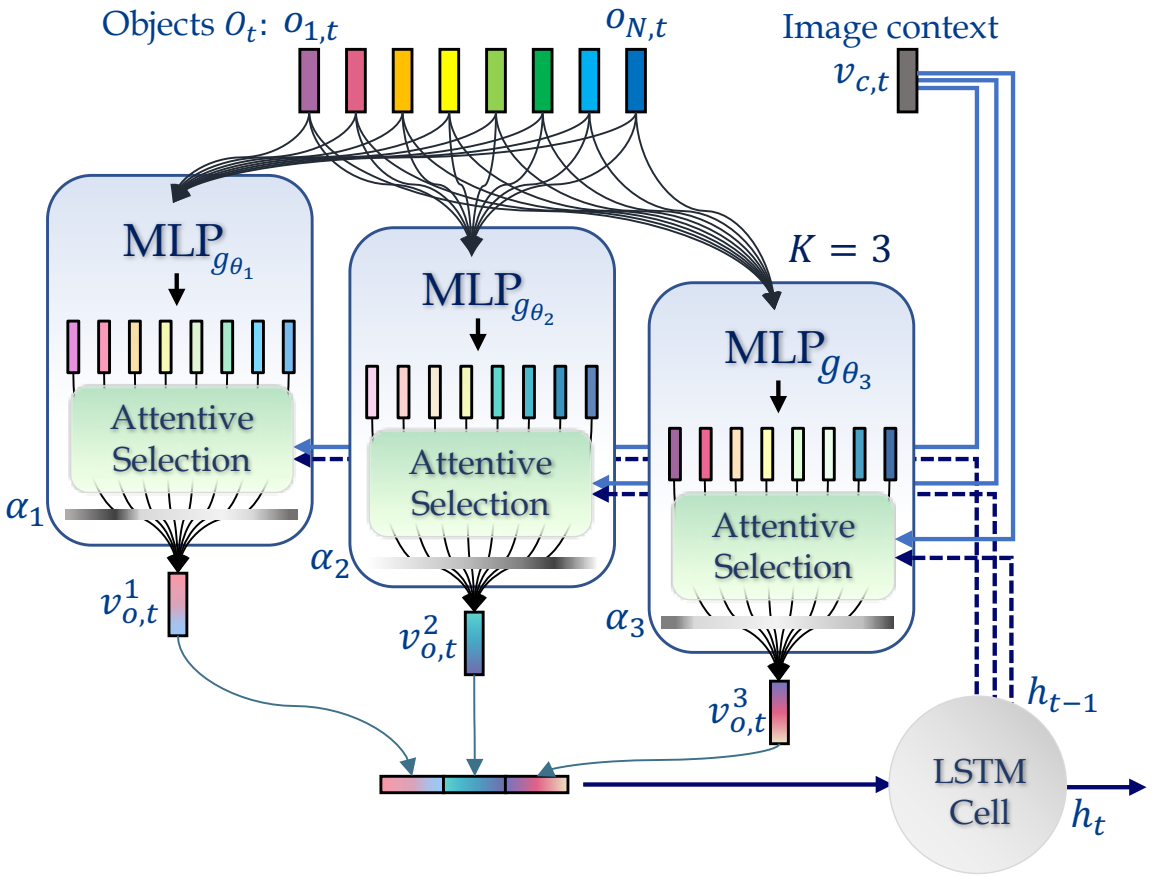

Figure 5.4: **Recurrent Higher-Order Interaction** module dynamically selects K groups of arbitrary objects with detected inter-object relationships via learnable attention mechanism. This attentive selection module uses the overall image context representation $v_{c,t}$, current set of (projected) objects O_t, and previous object interactions h_{t-1} to generate k^{th} weights α_k for k^{th} selections. The higher-order interaction between groups of selected objects is then modeled via concatenation and the following LSTM cell.

where $W_{h_k} \in \mathbb{R}^{d_\theta \times d_h}$ and $W_{c_k} \in \mathbb{R}^{d_\theta \times d_{v_{c,t}}}$ are learned weights for h_{t-1} and $v_{c,t}$, d_θ is the dimension of last fully-connected layer of g_{θ_k}, $X_k \in \mathbb{R}^{d_\theta \times N}$ is the input to k^{th} attention module, and $\sqrt{d_\theta}$ is a scaling factor, $\alpha_k \in \mathbb{R}^{N \times N}$ is the computed k^{th} attention. We omit the bias term for simplicity. The attended object feature at time t is then calculated as mean-pooling on weighted objects:

$$v^k_{o,t} = \overline{\alpha_k \left(g_{\theta_k}(O_t)\right)^\top} \tag{5.6}$$

where the output $v^k_{o,t}$ is a single feature vector representation which encodes the k^{th} object

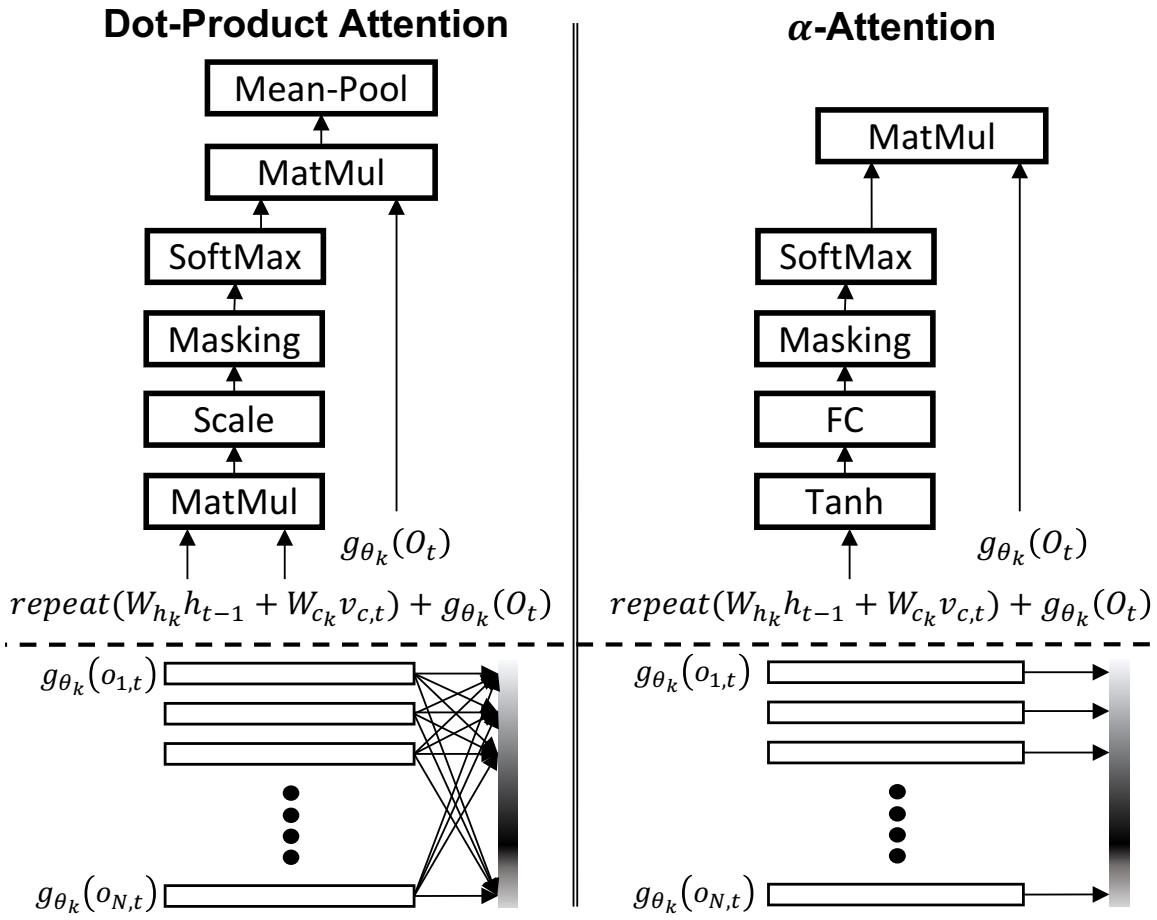

Figure 5.5: Attention modules: dot-product attention and α-attention. Both attention mechanisms take input from overall image representation $v_{c,t}$, current set of objects O_t, and previous object interactions h_{t-1} computed from LSTM cell at time $t-1$.

inter-relationships of a video frame at time t.

- α-attention. The α-attention uses the same input format as dot-product attention, but the attention is computed using a *tanh* function and a fully-connected layer. The attended object feature at time t can be calculated as a convex combination:

$$\alpha_k = softmax({w_k}^\top tanh(X_k)) \quad and \quad v_{o,t}^k = \sum_n \alpha_{k_n}(g_{\theta_k}(o_{n,t})) \tag{5.7}$$

where $w_k \in \mathbb{R}^{d_\theta}$ is a learned weight, and $\alpha_k \in \mathbb{R}^{1\times N}$ is the computed k^{th} attention. The

attended object feature at time t is then calculated as a convex combination:

$$v_{o,t}^{k} = \sum_{n} \alpha_{k_n}(g_{\theta_k}(o_{n,t})) \tag{5.8}$$

We use the α-attention as a baseline to show how considering the inter-relationships of objects (dot-product attention) can further improve the accuracy when ROIs are selected separately.

Finally, for both attention mechanisms, the selected object candidates $v_{o,t}^{k}$ are then concatenated and used as the input to a LSTM cell. The output $v_{oi,t}$ is then defined as the higher-order object interaction representation at current time t.

$$v_{oi,t} = LSTMCell(v_{o,t}^{1} \| v_{o,t}^{2} \| ... \| v_{o,t}^{K}) \tag{5.9}$$

where $\|$ denotes concatenation between feature vectors. The last hidden state of the LSTM cell $h_T = v_{oi,T}$ is the representation of overall object interactions for the entire video.

Note that by concatenating selected inter-object relationships into a single higher-order interaction representation, the selective attention module tends to select different groups of inter-relationships, since concatenating duplicate inter-relationships does not provide extra information and will be penalized. For an analysis of what inter-relationships are selected, please refer to Sec. 5.5.1.

5.1.4 Late fusion of coarse and fine

Finally, the attended context information v_c obtained from the image representation provides coarse-grained understanding of the video, and the object interactions discovered through the video sequences $v_{oi,T}$ provide fine-grained understanding of the video. We concatenate them as the input to the last fully-connected layer, and train the model jointly

to make a final action prediction.

$$p(y) = softmax(W_p(v_c \| v_{oi,T}) + b_p) \tag{5.10}$$

where $W_p \in \mathbb{R}^{d_y \times (d_{v_c} + d_{v_{oi,T}})}$ and $b_p \in \mathbb{R}^{d_y}$ are learned weights and biases.

5.2 Fine-grained Video Captioning

We now describe how SINet can be extended from sequence-to-one to a sequence-to-sequence problem for video captioning — **SINet-Caption**. Our goal in providing fine-grained information for video captioning is that, for each prediction of the word, the model is aware of the past generated word, previous output, and the summary of the video content. At each word generation, it has the ability to selectively attend to various parts of the video content in both space and time, as well as to the detected object interactions.

Our SINet-Caption is inspired by prior work using hierarchical LSTM for captioning tasks [141, 64], and we extend and integrate it with SINet so that the model can leverage the detected higher-order object interactions. We use a two-layered LSTM integrated with the coarse- and fine-grained information, as shown in Figure 5.6. The two LSTM layers are: Attention LSTM and Language LSTM. The Attention LSTM identifies which part of the video in spatiotemporal feature space is needed for Language LSTM to generate the next word. Different from prior work, which applied attention directly over all image patches in the entire video [66], *i.e.,* attended to objects individually, our attentive selection module attends to object interactions while considering their temporal order.

Attention LSTM. The Attention LSTM fuses the previous hidden state output of Language LSTM $h^2_{t_w-1}$, overall representation of the video, and the input word at time $t_w - 1$ to generate the hidden representation for the following attention module. Formally, the input

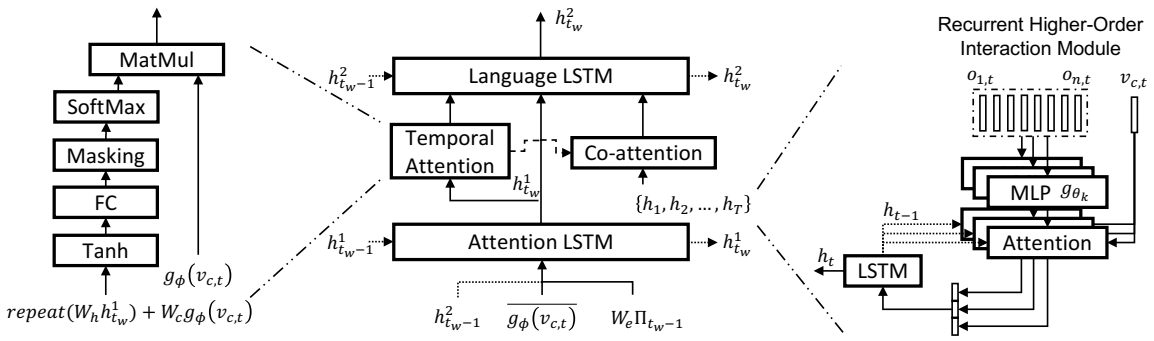

Figure 5.6: Overview of the proposed SINet-Caption for video captioning. The Attention LSTM with α-attention is used to selectively attend to temporal video frame features. The computed temporal attention is then used to attend to temporal object interactions $\{h_1, h_2, ..., h_T\}$ (see Figure 5.4). Concatenation of the outputs of Attention LSTM, attended video frame feature, and attended object interactions is then used as input for language decoder LSTM.

to Attention LSTM can be defined as:

$$x^1_{t_w} = h^2_{t_w-1} \parallel \overline{g_\phi(V_c)} \parallel W_e \Pi_{t_w-1} \tag{5.11}$$

where $\overline{g_\phi(V_c)}$ is the projected and mean-pooled image features, g_ϕ is a MLP with parameters ϕ, $W_e \in \mathbb{R}^{E \times \Sigma}$ is a word embedding matrix for a vocabulary of size Σ, and Π_{t_w-1} is one-hot encoding of the input word at time $t_w - 1$. Note that t is the video time, and t_w is the timestep for each word generation.

Temporal attention module. We adapt the regular soft-attention same α-attention module as shown in Figure 5.5 to attend over projected image features $g_\phi(V_c)$. The two types of input for this temporal attention module are from outputs of the Attention LSTM and projected image features.

$$X_a = repeat(W_h h^1_{t_w}) + W_c g_\phi(V_c) \tag{5.12}$$

where $h^1_{t_w}$ is the output of Attention LSTM, $W_h \in \mathbb{R}^{d_\phi \times d_{h^1_{t_w}}}$ and $W_c \in \mathbb{R}^{d_\phi \times d_\phi}$ are learned weights for $h^1_{t_w}$ and $g_\phi(V_c)$. d_ϕ is the dimension of the last FC layer of g_ϕ.

Co-attention We directly apply the temporal attention obtained from image features on

object interaction representations $\mathbf{h} = \big(h_1, h_2, ..., h_T\big)$ (see Sec 5.1.2 for details).

Language LSTM. Finally, the Language LSTM takes in input which is the concatenation of output of the Attention LSTM $h^1_{t_w}$, attended video representation $\hat{v}_{c,t_w}$, and co-attended object interactions $\hat{h}_{t_w}$ at timestep t_w.

$$x^2_{t_w} = h^1_{t_w} \parallel \hat{v}_{c,t_w} \parallel \hat{h}_{t_w} \tag{5.13}$$

The output of Language LSTM is then used to generate each word, which is a conditional probability distribution defined as:

$$p(y_{t_w}|y_{1:t_w-1}) = softmax(W_p h^2_{t_w}) \tag{5.14}$$

where $y_{1:t_w-1}$ is a sequence of outputs $(y_1, ..., y_{t_w-1})$ and $W_p \in \mathbb{R}^{\Sigma \times d_{h^2_{t_w}}}$ is learned weights for $h^2_{t_w}$. All bias terms are omitted for simplicity.

5.3 Datasets and Implementations

5.3.1 Datasets:

Video understanding can be classified into two tasks: sequence-to-one and sequence-to-sequence. In the following, we first describe the datasets used for these two problems followed by the implementation details and training procedures of our SINet.

Kinetics dataset: To evaluate SINet on a sequence-to-one problem for video, we use the Kinetics dataset for action recognition [132]. The Kinetics dataset contains 400 human action classes and has approximately 300k video clips (833 video hours). Most importantly, different from previous datasets which mostly cover sports actions [136, 135, 134], Kinetics includes human-object interactions and human-human interactions. We sampled videos at 1 FPS only, as opposed to sampling at 25 FPS reported for Kinetics [132].

ActivityNet Captions dataset: To evaluate SINet-Caption on a sequence-to-sequence

problem for video, we use ActivityNet Captions for video captioning. The ActivityNet Captions dataset contains 20k videos and has total of 849 video hours with 100K total descriptions. To demonstrate our proposed idea, we focus on providing fine-grained understanding of the video to describe video events with natural language, as opposed to identifying the temporal proposals. We thus use the ground truth temporal segments and treat each temporal segment independently. We use this dataset over others because ActivityNet Captions is action-centric, as opposed to object-centric [133]. This fits our goal of detecting higher-order object interactions for understanding human actions. All sentences are capped to be a maximum length of 30 words. We sample predictions using beam search of size 5 for captioning. While the previous work sample C3D features every 8 frames [133], we only sampled video at maximum 1 FPS. Video segments longer than 30 secs. are evenly sampled at maximum 30 samples.

5.3.2 Implementation Details:

We now discuss how to extract image and object features for both Kinetics and ActivityNet Captions.

Image feature: We fine-tune a pre-trained ResNeXt-101 [142] on Kinetics sampled at 1 FPS (approximately 2.5 million images). We use SGD with Nesterov momentum as the optimizer. The initial learning rate is $1e-4$ and drops by 10x when validation loss saturates for 5 epochs. The weight decay is $1e-4$ and the momentum is 0.9, and the batch size is 128. We use standard data augmentation by randomly cropping and horizontally flipping video frames during training. When extracting image features, the smaller edge of the image is scaled to 256 pixels and we crop the center of the image as input to the fine-tuned ResNeXt-101. Each image feature is a 2048-d feature vector.

Object feature: We generate the object features by first obtaining the coordinates of ROIs from a Deformable R-FCN [143] (pre-trained on MS-COCO) with ResNet-101 [144] as backbone architecture. We set the IoU threshold for NMS to be 0.2. Empirically, we

Table 5.1: Prediction accuracy on the Kinetics validation set. All of our results use only RGB videos sampled at 1 FPS. Maximum number of objects per frame is set to be 30.

Method	Top-1	Top-5
I3D[2](25 FPS) [17] (test)	71.1	89.3
TSN (Inception-ResNet-v2) (2.5 FPS) [14, 145]	73.0	90.9
Ours (1 FPS)		
Img feature + LSTM (baseline)	70.6	89.1
Img feature + temporal SDP-Attention	71.1	89.6
Obj feature (mean-pooling)	72.2	90.2
Img + obj feature (mean-pooling)	73.1	91.1
SINet (α-attention)	73.9	91.5
SINet (dot-product attention)	**74.2**	**91.7**

found that it is important to maintain a balance of image and object features, especially when image features were obtained from a network which was fine-tuned on the target dataset. Thus, for each of the ROIs, we extract features using coordinates and adaptive max-pooling from the same model (ResNeXt-101) that was fine-tuned on Kinetics. The resulting object feature for each ROI is a 2048-d feature vector. ROIs are ranked according to their ROI scores. We select top 30 objects for Kinetics and top 15 for ActivityNet Captions. Note that we have a varied number of ROIs for each video frame, and video length can also be different. We do not use the object class information since we may miss some of the objects that were not detected, due to the cross-domain problem. For the same reason, the bounding-box regression process is not performed here since we do not have the ground-truth bounding boxes.

Training: We train SINet and SINet-Caption with ADAM optimizer. The initial learning rate is set to $1e-5$ for Kinetics and $1e-3$ for ActivityNet Captions. Both learning rates automatically drop by 10x when validation loss is saturated. The batch sizes are 64 and 32 respectively for Kinetics and ActivityNet Captions.

5.4 Experimental Results

5.4.1 Action Recognition on Kinetics

[2]Results obtained from https://github.com/deepmind/kinetics-i3d

Table 5.2: Comparison of pairwise (or triplet) object interaction with the proposed higher-order object interaction with dot-product attentive selection method on Kinetics. The maximum number of objects is set to be 15. FLOP is calculated per video. For details on calculating FLOP, please refer to Sec. 5.5.7.

Method	Top-1	Top-5	FLOP (e^9)
Obj (mean-pooling)	73.1	90.8	1.9
Obj pairs (mean-pooling)	73.4	90.8	18.3
Obj triplet (mean-pooling)	72.9	90.7	77.0
SINet ($K = 1$)	73.9	91.3	2.7
SINet ($K = 2$)	74.2	91.5	5.3
SINet ($K = 3$)	**74.2**	**91.7**	8.0

Does temporal SDP-Attention help? Several studies have pointed out that using temporal mean-pooling or LSTMs may not be the best method to aggregate the sequence of image representations for videos [14, 140, 15]. To overcome this issue, we use temporal SDP-Attention instead of LSTM. As we can see from Table 5.1, using temporal SDP-Attention has proven to be superior to traditional LSTM and already performs comparably with 3D ConvNet that uses a much higher video sampling rate.

Does object interaction help? We first evaluate how much higher-order object interactions can help in identifying human actions. Considering mean-pooling over the object features to be the simplest form of object interaction, we show that mean-pooling over the object features per frame and using LSTM for temporal reasoning has already outperformed single compact image representations, which is currently the trend for video classification methods. Directly combining image features with temporal SDP-Attention and object features over LSTM further reaches 73.1% top-1 accuracy. This already outperforms the state-of-the-art TSN [145] method using a deeper ConvNet with a higher video sampling rate. Beyond using mean-pooling as the simplest form of object interaction, our proposed method to dynamically discover and model higher-order object interactions further achieved 74.2% top-1 and 91.7% top-5 accuracy. The selection module with dot-product attention, in which we exploit the inter-relationships between objects within the same group, outperforms α-attention where the inter-relationships are ignored.

Table 5.3: METEOR, ROUGE-L, CIDEr-D, and BLEU@N scores on the ActivityNet Captions test and validation set. All methods use ground truth proposal except LSTM-A_3 [146]. Our results with ResNeXt spatial features use videos sampled at maximum 1 FPS only.

Method	B@1	B@2	B@3	B@4	ROUGE-L	METEOR	CIDEr-D
Test set							
LSTM-YT [62] (C3D)	18.22	7.43	3.24	1.24	-	6.56	14.86
S2VT [61] (C3D)	20.35	8.99	4.60	2.62	-	7.85	20.97
H-RNN [66] (C3D)	19.46	8.78	4.34	2.53	-	8.02	20.18
S2VT + full context [133] (C3D)	26.45	13.48	7.21	3.98	-	9.46	24.56
LSTM-A_3 + policy gradient + retrieval [146] (ResNet + P3D ResNet [18])	-	-	-	-	-	12.84	-
Validation set (Avg. 1st and 2nd)							
LSTM-A_3 (ResNet + P3D ResNet) [146]	17.5	9.62	5.54	**3.38**	13.27	7.71	16.08
LSTM-A_3 + policy gradient + retrieval [146] (ResNet + P3D ResNet [18])	17.27	9.70	5.39	3.13	14.29	8.73	14.75
SINet-Caption — img (C3D)	17.18	7.99	3.53	1.47	18.78	8.44	38.22
SINet-Caption — img (ResNeXt)	18.81	9.31	4.27	1.84	20.46	9.56	43.12
SINet-Caption — obj (ResNeXt)	19.07	9.48	4.38	1.92	20.67	9.56	44.02
SINet-Caption — img + obj — no co-attention (ResNeXt)	**19.93**	9.82	**4.52**	2.03	21.08	9.79	44.81
SINet-Caption — img + obj — co-attention (ResNeXt)	19.78	**9.89**	**4.52**	1.98	**21.25**	**9.84**	**44.84**

Does attentive selection help? Prior work on visual relationships and VQA concatenate pairwise object features for detecting object relationships. In this experiment, we compare the traditional way of creating object pairs or triplets with our proposed attentive selection method. We use temporal SDP-Attention for image features, and dot-project attention for selecting object interactions. As shown in Table 5.2, concatenating pairwise features marginally improves over the simplest form of object interactions while increasing the computational cost drastically. By further concatenating three object features, the space for meaningful object interactions becomes so sparse that it instead reduced the prediction accuracy, and the number of operations (FLOP) further increases drastically. On the other hand, our attentive selection method can improve upon these methods while saving significant computation time. Empirically, we also found that reducing the number of objects per frame from 30 to 15 yields no substantial difference on prediction accuracy. This indicates that the top 15 objects with highest ROI score are sufficient to represent fine-grained details of the video. For detailed qualitative analysis of how objects are selected at each timestep and how SINet reasons over a sequence of object interactions, please see Sec. 5.5.1.

We are aware of that integrating optical flow or audio information with RGB video can further improve the action recognition accuracy [14, 17]. We instead focus on modeling

object interactions for understanding video in a fine-grained manner, and we consider other modalities to be complementary to our higher-order object interactions.

5.4.2 Video Captioning on ActivityNet Captions

We focus on understanding human actions for video captioning rather than on temporal proposals. Hence, we use ground truth temporal proposals for segmenting the videos and treat each video segment independently. All methods in Table 5.3 use ground truth temporal proposal, except LSTM-A_3 [146]. Our performances are reported with four language metrics, including BLEU [147], ROUGH-L [148], METEOR [149], and CIDEr-D [150].

For fair comparison with prior methods using C3D features, we report results with both C3D and ResNeXt spatial features. Since there is no prior result reported on the validation set, we compare against LSTM-A_3 [146] which reports results on the validation and test sets. This allows us to indirectly compare with methods reported on the test set. As shown in Table 5.3, while LSTM-A_3 clearly outperforms other methods on the test set with a large margin, our method shows better results on the validation sets across nearly all language metrics. We do not claim our method to be superior to LSTM-A_3 because of two fundamental differences. First, they do not rely on ground truth temporal proposals. Second, they use features extracted from an ResNet fine-tuned on Kinetics and another P3D ResNet [18] fine-tuned on Sports-1M, whereas we use a ResNeXt-101 fine-tuned on Kinetics sampled at maximum 1 FPS. Utilizing more powerful feature representations has been proved to improve the prediction accuracy by a large margin on video tasks. This also corresponds to our experiments with C3D and ResNeXt features, where the proposed method with ResNeXt features perform significantly better than C3D features.

Does object interaction help? SINet-Caption without any object interaction has already outperformed prior methods reported on this dataset. Additionally, by introducing an efficient selection module for detecting object interactions, SINet-Caption further improves across nearly all evaluation metrics, with or without co-attention. We also observed that in-

troducing the co-attention from image features constantly shows improvement on the first validation set but having separate temporal attention for object interaction features show better results on second validation set (please see Sec. 5.5.6 for results on each validation set).

5.5 Discussions and Qualitative Analysis

5.5.1 Qualitative analysis on Kinetics

To further validate the proposed method, we qualitatively show how the SINet selectively attends to various regions with relationships and interactions across time. We show several examples in Figure 5.9, 5.10, and 5.11. In each of the figure, the top row of each video frame has generally multiple ROIs with three colors: red, green, and blue. ROIs with the same color indicates that there exist inter-relationships. We then model the interaction between groups of ROIs across different colors. The color of each bounding box is weighted by the attention generated by the proposed method. Thus, if some ROIs are not important, they will have smaller weights and will not be shown on the image. The same weights are then used to set the transparent ratio for each ROI. The brighter the region is, the more important the ROI is.

Focus on object semantics. Recent state-of-the-art methods for action recognition rely on single compact representation of the scene. We show that the proposed SINet can focus on the details of the scene and neglect the visual content that maybe irrelevant such as the background information. For example, in Figure 5.9, the model constantly focus on the rope above the water and the person riding on wakeboard. The same goes for Figure 5.10. The background scenes with ice and snow are ignored throughout the video since it's ambiguous and easy to be confused with other classes involve snow in the scene.

Adjustable inter-relationships selection. We notice that our SINet tends to explore the whole scene early in the video, i.e. the attentions tend to be distributed to the ROIs that cover large portion of the video frame, and the attentions become more focused after this

exploration stage.

5.5.2 Qualitative analysis on ActivityNet Captions

In addition to the qualitative analysis on action recognition task, we now present the analysis on video captioning. Several examples are shown in Figure 5.12, 5.13, and 5.14. At each word generation step, the SINet-Caption uses the weighted sum of the video frame representations and the weighted sum of object interactions at corresponding timesteps (co-attention). Note that, since we aggregate the detected object interactions via the LSTM cell through time, the feature representation of the object interactions at each timestep can be seen as a fusion of interactions at the present and past time. Thus, if temporal attention has highest weight on $t = 3$, it may actually attend to the interaction aggregated from $t = 1$ to $t = 3$. Nonetheless, we only show the video frame with highest temporal attention for convenience. We use red and blue to represent the two selected sets of objects ($K = 2$).

In each of the figures, the video frames (with maximum temporal attention) at different timesteps are shown along with each word generation. All ROIs in the top or bottom images are weighted with their attention weights. In the top image, ROIs with weighted bounding box edges are shown, whereas, in the bottom image, we set the transparent ratio equal to the weight of each ROI. The brighter the region is, the more important the ROI is. Therefore, less important ROIs (with smaller attention weights) will disappear in the top image and be completely black in the bottom image. When generating a word, we traverse the selection of beam search at each timestep.

As shown in Figure 5.12, we can see that the SINet-Caption can successfully identify the person and the wakeboard. These selections of the two most important objects imply that the person is riding on the wakeboard — water skiing. We also observe that, in Figure 5.13, the proposed method focuses on the bounding boxes containing both person and the camel. Suggesting that this is a video for people sitting on a camel. However, it failed to identify that there are in fact multiple people in the scene and there are two camels. On the

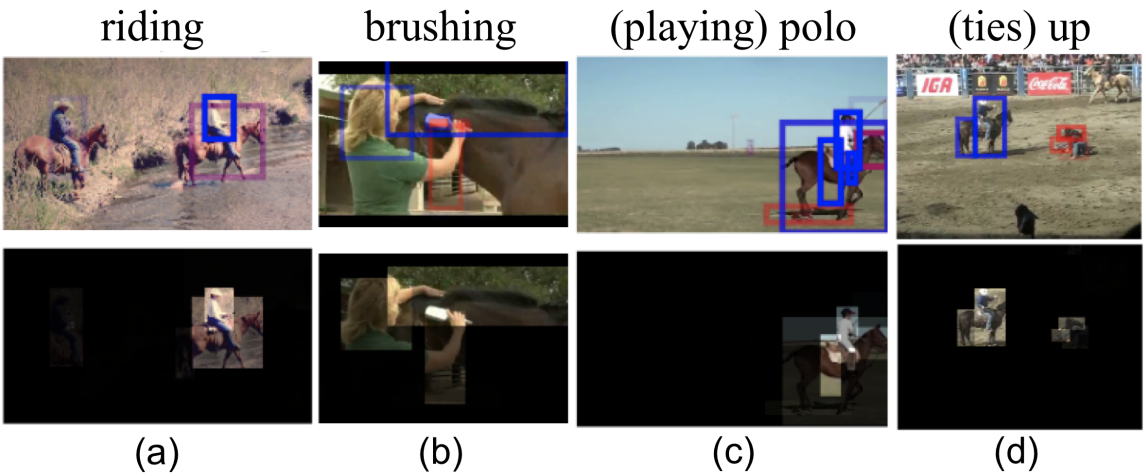

Figure 5.7: What interactions (verb) learned for video captioning. We verify how the SINet-Caption distinguishes various type of interactions with a common object - *horse*. (a) People are *riding* horses. (b) A woman is *brushing* a horse. (c) People are playing *polo* on a field. (d) The man *ties* up the calf.

other hand, the SINet-Caption is able to identify the fact that there are two persons playing racquetball in Figure 5.14.

5.5.3 Distinguish interactions when common objects presented

A common problem with the state-of-the-art captioning models is that they often lack the understanding of the relationships and interactions between objects, and this is oftentimes the result of dataset bias. For instance, when the model detects both person and a horse. The caption predictions are very likely to be: A man is riding on a horse, regardless whether if this person has different types of interactions with the horse.

We are thus interested in finding out whether if the proposed method has the ability to distinguish different types of interactions when common objects are presented in the scene. In Figure 5.7, each video shares a common object in the scene - *horse*. We show the verb (interaction) extracted from a complete sentence as captured by our proposed method.

- People are *riding* horses.
- A woman is *brushing* a horse.

- People are playing *polo* on a field.
- The man *ties* up the calf.

While all videos involve horses in the scene, our method successfully distinguishes the interactions of the human and the horse.

5.5.4 Discussion on ActivityNet Captions

We observed that while higher-order object interactions did contribute to higher performance on ActivityNet, the contributions were not as significant as when applied to the Kinetics dataset (quantitatively or qualitatively). We hereby discuss some potential reasons and challenges on applying SINet-Caption on the ActivityNet Captions dataset.

Word by word caption generation. In line with the work from question-answering, machine translation, and captioning, we generate a language sentence describing a video one word after another. At each word generation step, the SINet-Caption uses the last generated word, video frame representations, and their corresponding object interactions. As we can see from both qualitative results from Kinetics and ActivityNet Captions, our proposed method is able to identify the interactions within a very few video frames. However, taking Figure 5.13 as an example, at the first word "a", our model has already successfully selected the persons (both in light blue and red) on top of the camel (bright red). Yet, during the following caption generation, the SINet-Caption was *forced* to look at the visual content again and again. Introducing the gated mechanism [151] may mitigate this issue, but our preliminary results do not show improvement. Further experiments toward this direction may be needed.

Semantically different captions exist. Each video in the ActivityNet Captions dataset consists of 3.65 (average) different temporal video segments and their own ground truth captions [133]. These video captions have different semantic meanings but oftentimes share very similar video content, i.e. the same/similar video content has several different

ground truth annotations. As a result, it may create confusion during the training of the model. Again, taking Figure 5.13 as an example, we observed that the SINet-Caption often focuses on the person who leads the camels ($t = 1, 3, 15$). We conjecture that this is due to the fact that, within the same video, there exists another video segment with annotation: *A short person that is leading the camels turns around.* Although within the same video content, one of the ground truth focuses on the persons sitting on the camels, another ground truth focuses on the person leading the camels. This seems to be the reason why the trained network focuses on that particular person. Based on this observation, we believe that future work in re-formulating these semantically different annotations of similar video content for network training is needed, and perhaps it may be a better way to fully take advantage of fine-grained object interactions detected from SINet-Caption. One possibility will be associating semantically different video captions with different region-sequences within a video [70].

5.5.5 Performance improvement analysis on Kinetics

The proposed SINet ($K = 3$) shows more than 5% improvement on top-1 accuracy in 136/400 classes and more than 10% improvement in 46 classes over baseline. We show the classes that were improved more than 10% on top-1 accuracy in Figure 5.8. In addition to these classes, the proposed SINet in modeling fine-grained interactions specifically improved many closely related classes.

- 7 classes related to **hair** that are ambiguous among each other: *braiding hair*, *brushing hair*, *curling hair*, *dying hair*, *fixing hair*, *getting a haircut*, and *washing hair*. We show 21% top-1 improvement on *washing hair*; 16% improvement on *getting a haircut*.

- 4 classes related to **basketball** require the model to identify how the basketball are being interacted. These classes are: *playing basketball*, *dribbling basketball*, *dunking*

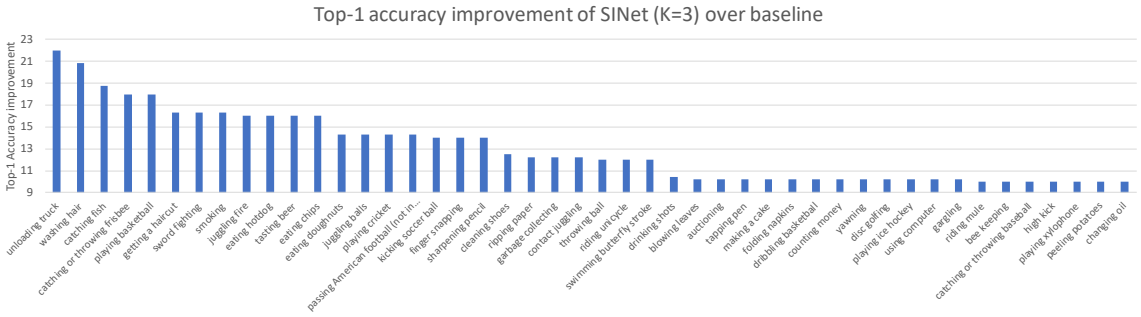

Figure 5.8: Top-1 accuracy improvement of SINet ($K = 3$) over baseline. 46/400 classes that are improved more than 10% are shown.

Table 5.4: METEOR, ROUGE-L, CIDEr-D, and BLEU@N scores on the ActivityNet Captions 1st and 2nd validation set. All methods use ground truth temporal proposal, and out results are evaluated using the code provided in [133] with $tIoU = 0.9$. Our results with ResNeXt spatial features use videos sampled at maximum 1 FPS only.

Method	B@1	B@2	B@3	B@4	ROUGE-L	METEOR	CIDEr-D
1st Validation set							
SINet-Caption — img (C3D)	16.93	7.91	3.53	1.58	18.81	8.46	36.37
SINet-Caption — img (ResNeXt)	18.71	9.21	4.25	2.00	20.42	9.55	41.18
SINet-Caption — obj (ResNeXt)	19.00	9.42	4.29	2.03	20.61	9.50	42.20
SINet-Caption — img + obj — no co-attention (ResNeXt)	**19.89**	9.76	4.48	2.15	21.00	9.62	43.24
SINet-Caption — img + obj (ResNeXt)	19.63	**9.87**	**4.52**	**2.17**	**21.22**	**9.73**	**44.14**
2nd Validation set							
SINet-Caption — img (C3D)	17.42	8.07	3.53	1.35	18.75	8.41	40.06
SINet-Caption — img (ResNeXt)	18.91	9.41	4.28	1.68	20.49	9.56	45.05
SINet-Caption — obj (ResNeXt)	19.14	9.53	4.47	1.81	20.73	9.61	45.84
SINet-Caption — img + obj — no co-attention (ResNeXt)	**19.97**	9.88	**4.55**	**1.90**	21.15	**9.96**	**46.37**
SINet-Caption — img + obj (ResNeXt)	19.92	**9.90**	4.52	1.79	**21.28**	9.95	45.54

basketball, and *shooting basketball*. We observed 18%, 10%, 6%, and 8% improvement respectively.

- Among 3 related to **juggling** actions: *juggling fire*, *juggling balls*, and *contact juggling*. We obtained 16%, 14%, and 13% improvement respectively.

- Our model significantly improved the **eating** classes, which are considered to be the hardest [132], because they require distinguishing what is being eaten (interacted). We show improvement among all eating classes, including *eating hot dog*, *eating chips*, *eating doughnuts*, *eating carrots*, *eating watermelon*, and *eating cake*. We obtained 16%, 16%, 14%, 8%, 4%, and 4% improvement respectively.

Table 5.5: FLOPs calculation on Kinetics sampled at 1 FPS. The calculation is based on forward passing of one video.

Proposed method ($K = 2$)		FLOP	Object pairs		FLOP
Project obj features					
MLP $g_{\theta_k(o_{i,t})}$	15 x 2048 x 2048 x 2	0.13e9	MLP	105 x 4096 x 2048	0.9e9
	15 x 2048 x 2048 x 2	0.13e9		105 x 2048 x 2048	0.4e9
	15 x 2048 x 2048 x 2	0.13e9		105 x 2048 x 2048	0.4e9
Recurrent unit					
Recurrent HOI (SDP-Attention)					
$W_h h_{t-1}$	2048 x 2048 x 2	8.4e6			
$W_c v_{c,t}$	2048 x 2048 x 2	8.4e6			
MatMul	15 x 15 x 2048 x 2	0.9e6			
MatMul	15 x 15 x 2048 x 2	0.9e6			
LSTM Cell	8 x 2 x 2 x 2048 x 2048	134.2e6	LSTM Cell	8 x 2 x 2048 x 2048	67e6
Total					
timesteps ($T = 10$)	10 x (MLP + Recurrent)	5.3e9		10 x (MLP + Recurrent)	18.3e9

5.5.6 ActivityNet Captions on 1st and 2nd val set

We report the performance of SINet-Caption on the 1st and the 2nd validation set in Table 5.4. We can see that using fine-grained (higher-order) object interactions for caption generation consistently shows better performance than using coarse-grained image representation, though the difference is relatively minor compared to the results on Kinetics. We discuss the potential reasons in Sec. 5.5.2. Combining both coarse- and fine-grained improve the performance across all evaluation metrics. Interestingly, using co-attention on detected object interactions shows better performance on the 1st validation set but has similar performance on the 2nd validation set.

5.5.7 Model architecture and FLOP

We now describe the model architecture of the proposed recurrent higher-order module and how the FLOP is calculated.

SINet architecture. We first project the image representations $v_{c,t}$ to introduce learnable feature representations. The MLP g_ϕ consist of two sets of fully-connected layers each with batch normalization and ReLU. It maintains same dimension ($m = 2048$) of the input image feature. Thus, the coarse-grained representation of the video is a feature vector with

2048 dimension. Inside the Recurrent HOI module, each of the MLP g_{θ_k} has three sets of batch normalization layers, fully-connected layers, and ReLUs. In the experiments with two attentive selection module ($K = 2$), we set the dimension of the fully-connected layer to be 2048. The concatenation of $v^1_{o,t}$ and $v^2_{o,t}$ is then used as the input to the following LSTM cell. Empirically, we find out that it's important to maintain high dimensionality for the input to LSTM cell. We adjust the dimension of hidden layers in g_{θ_k} given the number of K, e.g. we reduce the dimension of the hidden layer if K increases. In this way, the inputs to LSTM cell have the same or similar feature dimension for fair experimental comparison. The hidden dimension of the LSTM cell is set to be 2048. Before concatenating the coarse- (v_c) and fine-grained ($v_{oi,T}$) video representations, we re-normalize the feature vector with batch normalization layer separately. The final classifier then projects the concatenated feature representation to 400 action classes.

SINet-Caption architecture. We first use a single fully-connected layer with batch normalization, dropout, and ReLU to project the pre-saved image features $v_{c,t}$. The g_ϕ maps the feature vector from 2048 to 1024. We use two attentive selection modules for video captioning task ($K = 2$). Each g_{θ_k} consist of a batch normalization, fully-connected layer, dropout layer, and a ReLU. It maps input object feature vector from 2048 to 512. The dropout ratio for both g_ϕ and g_{θ_k} are set to be 0.5. The concatenation of $v^1_{o,t}$ and $v^2_{o,t}$ is used as input to the LSTM cell inside Recurrent HOI module. The hidden dimension of this LSTM cell is set to be 1024. The dimension of word embedding is 512. We use ReLU and dropout layer after embedding layer with dropout ratio 0.25. The hidden dimension of both Attention LSTM and Language LSTM are set to be 512.

FLOP is computed per video and the maximum number of objects per frame is set to 15. We compare the computed FLOP with traditional object interactions by paring all possible objects. The results are shown in Table 5.5.

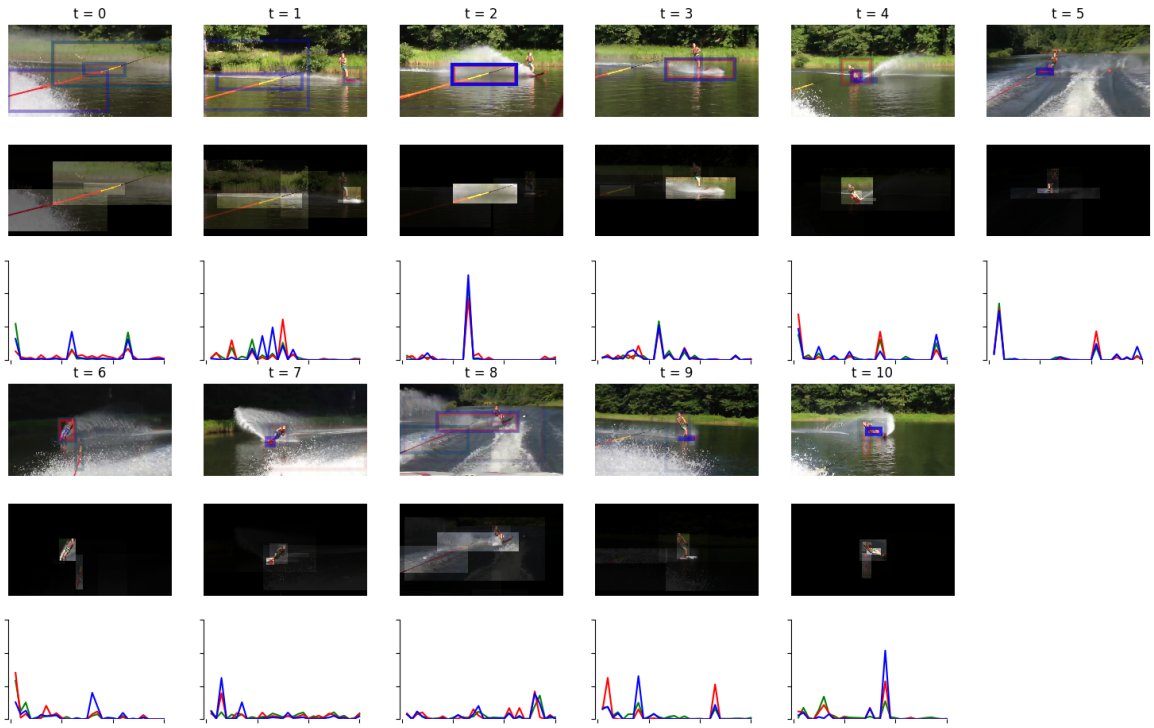

Figure 5.9: **Water skiing**: Our SINet is able to identify several object relationships and reasons these interactions through time: (1) the rope above the water (2) the wakeboard on the water (3) human riding on the wakeboard (4) rope connecting to the person on the wakeboard. From the distribution of three different attention weights (red, green, blue), we can also see that the proposed attention method not only is able to select objects with different inter-relationships but also can use a common object to discover different relationships around that object when needed. We observed that our method tends to explore the whole scene at the beginning of the video, and focus on new information that is different from the past. For example, while video frame at first few frames are similar, the model focus on different aspect of the visual representation.

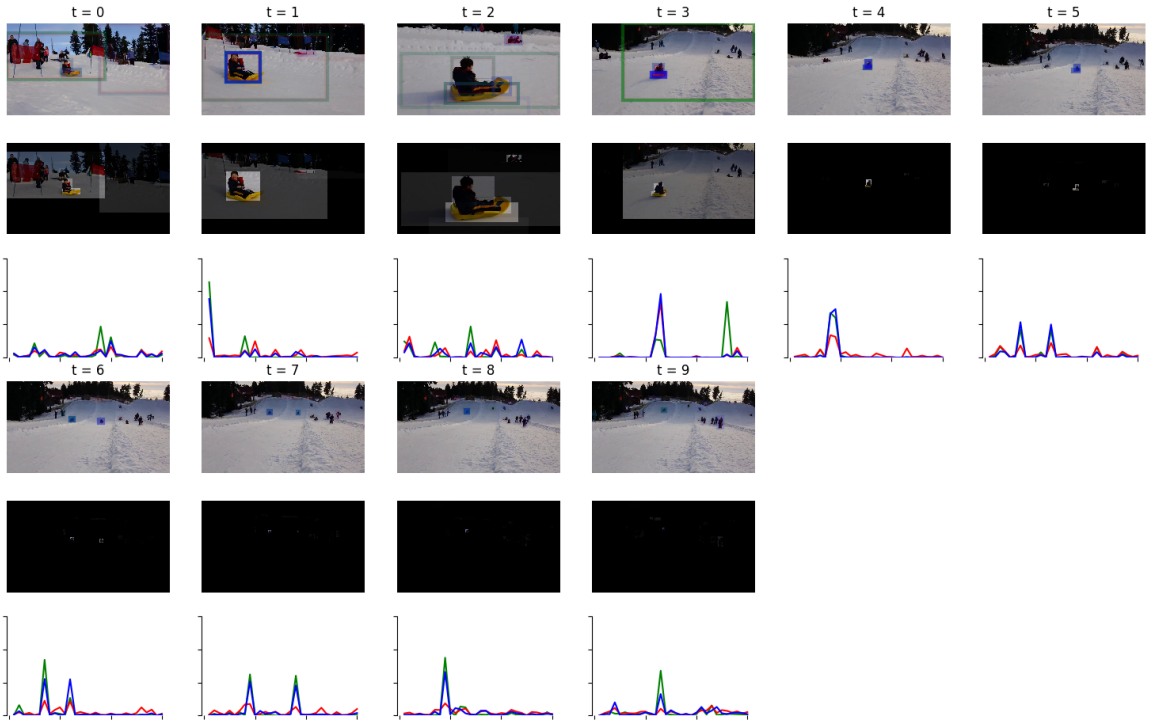

Figure 5.10: **Tobogganing**: Identifying *Tobogganing* essentially need three elements: toboggan, snow scene, and a human sitting on top. The three key elements are accurately identified and their interaction are highlighted as we can see from $t = 1$ to $t = 3$. Note that the model is able to continue tracking the person and toboggan throughout the whole video, even though they appear very small towards the end of the video. We can also noticed that our SINet completely ignore the background scene in the last several video frames as they are not informative since they can be easily confused by other 18 action classes involving snow and ice, e.g. *Making snowman*, *Ski jumping*, *Skiing crosscountry*, *Snowboarding*, etc.

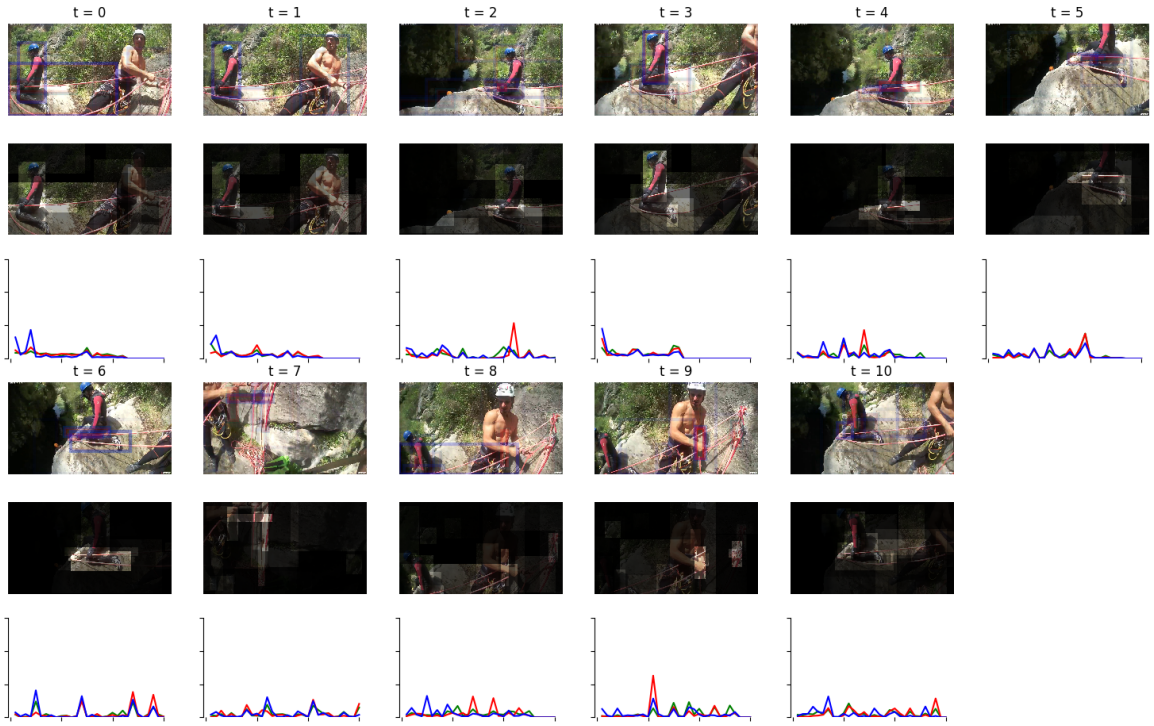

Figure 5.11: **Abseiling** is challenging since there are similar classes exist: *Climbing a rope*, *Diving cliff*, and *Rock climbing*, which involve ropes, rocks and cliffs. To achieve this, the model progressively identify the interactions and relationships like: human sitting the rock, human holding the rope, and the presence of both rope and rock. This information is proven to be sufficient for predicting *Abseiling* over other ambiguous action classes.

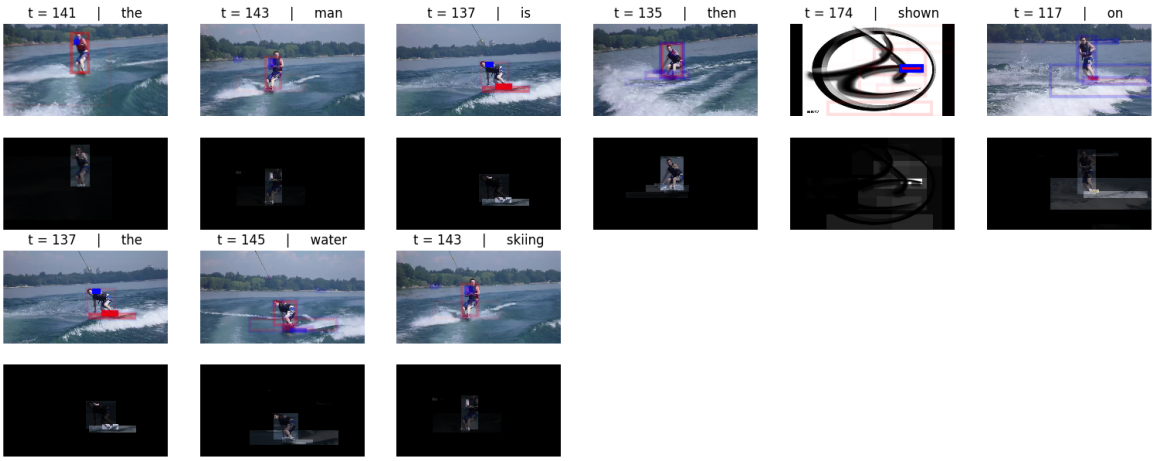

Figure 5.12: *The man is then shown on the water skiing.* We can see that the proposed SINet-Caption often focus on the person and the wakeboard, and most importantly it highlight the interaction between the two, i.e. the person steps on the wakeboard.

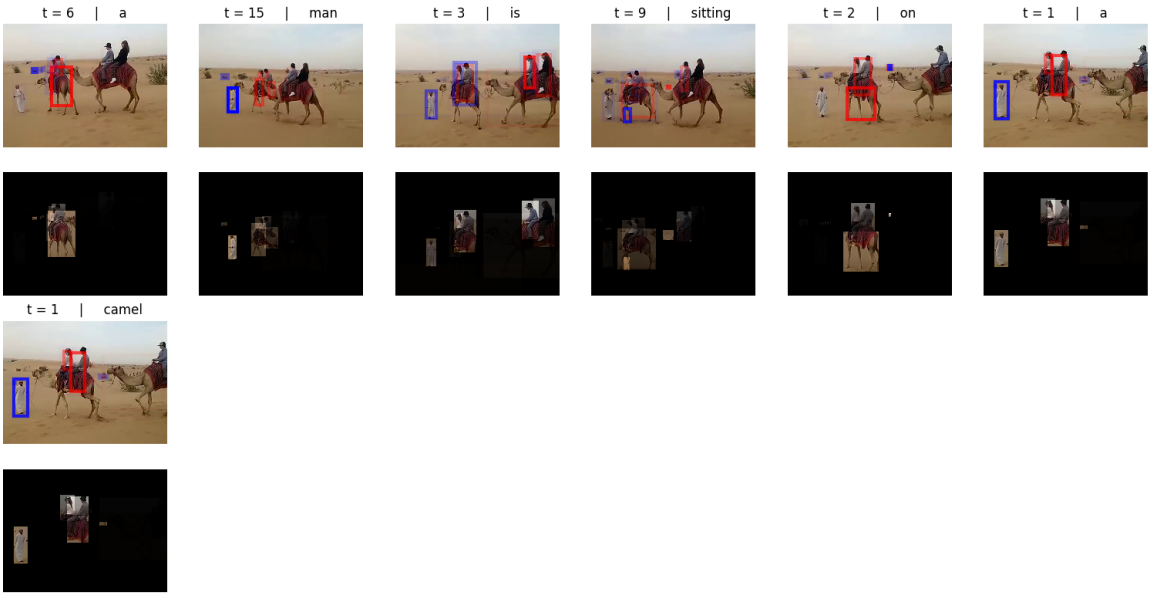

Figure 5.13: *A man is sitting on a camel.* The SINet-Caption is able to detect the ROIs containing both persons and the camel. We can also observe that it highlights both the ROIs for persons who sit on the camel and the camel itself at frame 3 and 9. However, the proposed method failed to identify that there are multiple people sitting on two camels. Furthermore, in some cases, it selects the person who leads the camels. This seems to be because the same video is also annotated with another caption focusing on that particular person: *A short person that is leading the camels turns around.*

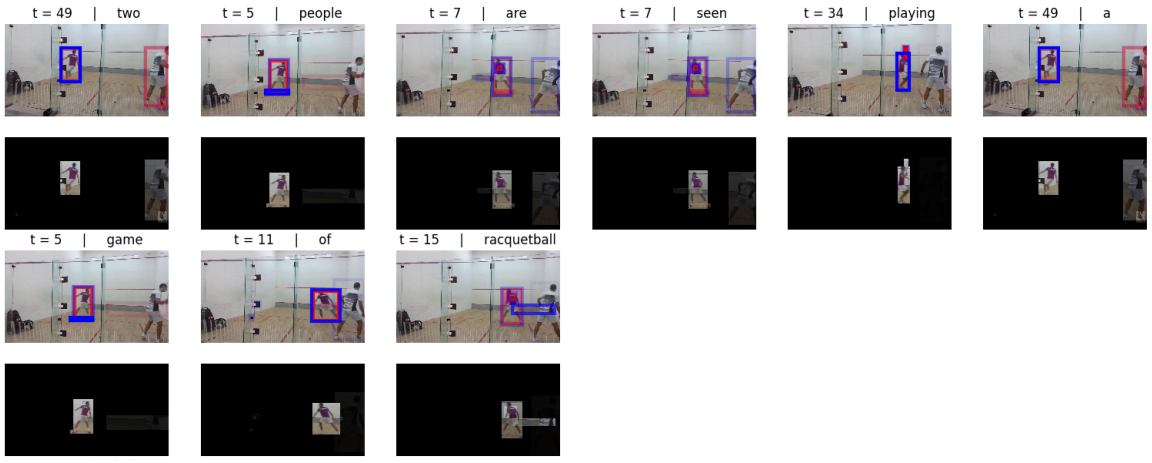

Figure 5.14: *Two people are seen playing a game of racquetball.* The SINet-Caption is able to identify that two persons are playing the racquetball and highlight the corresponding ROIs in the scene.

CHAPTER 6
GROUNDED VISUAL CAPTIONING WITHOUT LOCALIZATION SUPERVISION

In Chapter 5.2, we present a captioning model that is capable of leveraging the object-level relational reasoning for video understanding. However, visual captioning models are often not grounded on the captions they generated [20], which likely lead to the hallucination of objects that are not presented in the image or video [22], despite having high captioning accuracy. That is, they often do not correctly associate generated words with the appropriate image regions (*e.g.,* objects) in the scene, resulting in models that lack interpretability.

Several existing approaches have tried to improve the grounding of captioning models. One class of methods generate sentence *templates* with slot locations explicitly tied to specific image regions. These slots are then filled in by visual concepts identified by off-the-shelf object detectors [87]. Other methods have developed specific grounding or attention modules that aim to *attend* to the correct region(s) for generating visually groundable word. Such methods, however, rely on explicit supervision for optimizing the grounding or attention modules [20, 109] and require bounding box annotations for each visually groundable word.

In this chapter, we propose a novel cyclical training regimen that is able to significantly improve grounding performance without any grounding annotations. The key insight of our work is that current models use attention mechanisms conditioned on the hidden features of recurrent modules such as LSTMs, which leads to effective models with high accuracy but entangle grounding and decoding. Since LSTMs are effective at propagating information across the decoding process, the network does not necessarily need to associate particular decoded words with their corresponding image region(s). However, for a captioning model to be visually grounded, the model has to predict attentional weights without knowing the

word to localize.

Based on this insight, we develop a cyclical training regimen to force the network to ground individual decoded words: *decoding* $\rightarrow$ *localization* $\rightarrow$ *reconstruction*. Specifically, the model of the decoding stage can be any state-of-the-art captioning model; in this work, we follow GVD [109] to extend the widely used Up-Down model [86]. At the localization stage, each word generated by the first decoding stage is localized through a *localizer*, and the resulting grounded image region(s) are then used to reconstruct the ground-truth caption in the final stage. Both decoding and reconstruction stages are trained using a standard cross-entropy loss. Key to our method, both stages share the same decoder, thereby causing the localization stage to guide the decoder to improve its attention mechanism. Our method is simple and only adds a fully-connected layer to perform localization. During inference, we only use the (shared) decoder, thus we do not add any computational cost.

We benchmark our proposed method on the challenging Flickr30k Entities image captioning dataset [152] and the ActivityNet-Entities video captioning dataset [109] on both captioning and grounding performances. In addition to the existing grounding metric that calculate the grounding accuracy for each object class [109], we further include a grounding metric that compute grounding accuracy for each generated sentence. This new metric on each sentence removes the stringency of the original evaluation metric (as we discuss in Sec. 6.3) and provides an alternative way of measuring the grounding performance.

Despite the simplicity of our proposed method, we are able to significantly surpass prior unsupervised models quantitatively and qualitatively on both datasets. We achieve around 18% relative improvements in terms of bridging the gap between the unsupervised baseline and supervised methods on Flickr30k Entites and around 34% on ActivityNet-Entities. We further find that our method can even outperform the supervised method on infrequent words, owing to its self-supervised nature.

Contributions summary. We propose object re-localization as a form of self-supervision

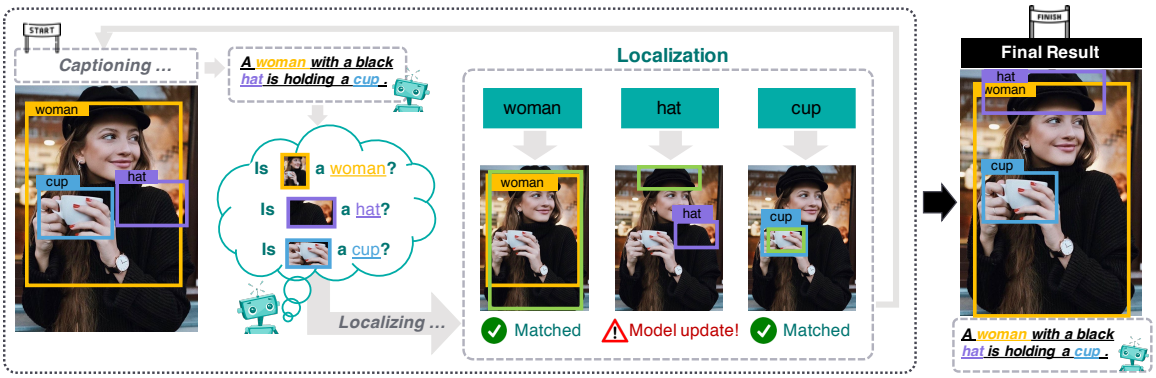

Figure 6.1: Visual captioning models are often not visually-grounded. As human, we perform localization to check whether the generated caption is visually-grounded. If the localized image region is incorrect, we update the model. However, without the ground-truth grounding annotation, how does the model know the localized region is incorrect? To overcome this issue, we propose to perform *localization* and *reconstruction* to regularize the captioning model to be visually-grounded without relying on the grounding annotations.

for grounded visual captioning and present a cyclical training regimen that re-generates sentences after re-localizing the objects conditioned on each word, implicitly imposing grounding consistency. We evaluate our proposed approach on both image and video captioning tasks. We show that the proposed training regime can boost grounding accuracy over a state-of-the-art baseline, enabling grounded models to be trained without bounding box annotations, while retaining high captioning quality across two datasets and various experimental settings.

6.1 Method

Notation. For a visual captioning task, we denote the input image as I (or input video as V) and the target sentence as S. Each image (or video) is represented by spatial feature map(s) extracted by a ResNet-101 model and a bag of regions obtained from Faster-RCNN [102] as $\boldsymbol{R} = [\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_N] \in \mathbb{R}^{d \times N}$. The target sentence is represented as a sequence of one-hot vectors $\boldsymbol{y}_t^* \in \mathbb{R}^s$, where T is the sentence length, $t \in 1, 2, ..., T$, and s is the dictionary size.

6.1.1 Baseline

We reimplemented the model used in GVD [109] without self-attention for region feature encoding [5, 126] as our baseline. It is an extension of the state-of-the-art Up-Down [86] model with the *grounding-aware region encoding* (see Sec. 6.2.3). Specifically, our baseline model uses two LSTM modules: Attention LSTM and Language LSTM. The Attention LSTM identifies which visual representation in the image is needed for the Language LSTM to generate the next word. It encodes the global image feature $\boldsymbol{v}_g$, previous hidden state output of the Language LSTM $\boldsymbol{h}_{t-1}^L$, and the previous word embedding e_{t-1} into the hidden state $\boldsymbol{h}_t^A$.

$$\boldsymbol{h}_t^A = LSTM_{Attn}([\boldsymbol{v}_g; \boldsymbol{h}_{t-1}^L; \boldsymbol{e}_{t-1}]), \quad \boldsymbol{e}_{t-1} = \boldsymbol{W}_e \boldsymbol{y}_{t-1}, \tag{6.1}$$

where $[;]$ denotes concatenation, and $\boldsymbol{W}_e$ are learned parameters. We omit the Attention LSTM input hidden and cell states to avoid notational clutter in the exposition.

The Language LSTM uses the hidden state $\boldsymbol{h}_t^A$ from the Attention LSTM to dynamically attend on the bag of regions $\boldsymbol{R}$ for obtaining visual representations of the image $\hat{\boldsymbol{r}}_t$ to generate a word y_t.

$$z_{t,n} = \boldsymbol{W}_{aa} tanh(\boldsymbol{W}_a \boldsymbol{h}_t^A + \boldsymbol{r}_n), \quad \boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{z}_t), \quad \hat{\boldsymbol{r}}_t = \boldsymbol{R}\boldsymbol{\alpha}, \tag{6.2}$$

where $\boldsymbol{W}_{aa}$ and $\boldsymbol{W}_a$ are learned parameters. The conditional probability distribution over possible output words $\boldsymbol{y}_t$ is computed as:

$$\boldsymbol{h}_t^L = LSTM_{Lang}([\hat{\boldsymbol{r}}_t, \boldsymbol{h}_t^A]), \quad p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}) = \text{softmax}(\boldsymbol{W}_o \boldsymbol{h}_t^L), \tag{6.3}$$

where $\boldsymbol{y}_{1:t-1}$ is a sequence of outputs $(\boldsymbol{y}_1, ..., \boldsymbol{y}_{t-1})$. We refer the Language LSTM and the output logit layer as the complete language decoder.

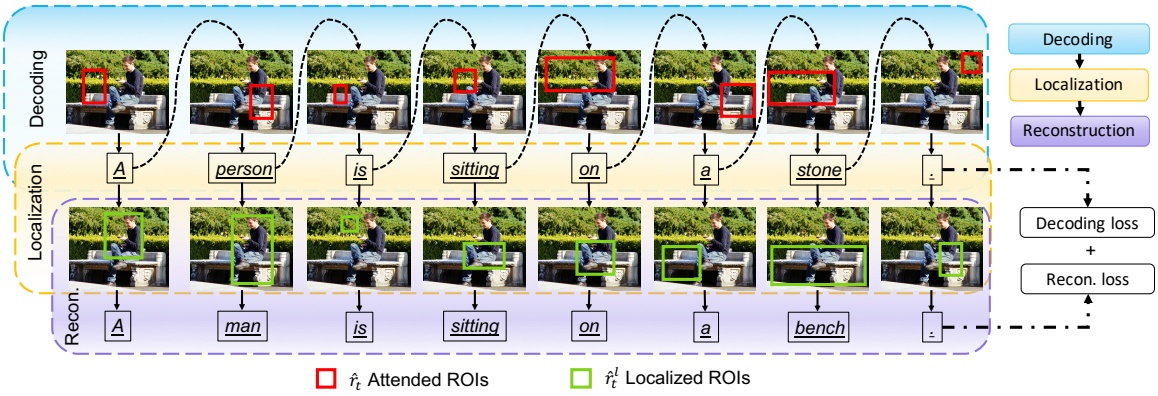

Figure 6.2: Proposed cyclical training regimen: *decoding* → *localization* → *reconstruction*. The decoder attends to the image regions and sequentially generate each of the output words. The localizer then uses the generated words as input to locate the image regions. Finally, the shared decoder during reconstruction stage uses the localized image regions to regenerate a sentence that matches with the ground-truth sentence.

6.1.2 Proposed Method Overview

Our goal is to enforce the generated caption to be visually grounded, *i.e.,* attended image regions correspond specifically to individual words being generated, *without* ground-truth grounding supervision. Towards this end, we propose a novel cyclical training regimen that is comprised of *decoding*, *localization*, and *reconstruction* stages, as illustrated in Figure 6.2.

The intuition of our method is that the baseline network is not forced to generate a correct correspondence between the attended objects and generated words, since the LSTMs can learn priors in the data instead of looking at the image or propagate information forward which can subsequently be used to generate corresponding words in future time steps. The proposed cyclical training regimen, in contrast, aims at enforcing visual grounding to the model by requiring the language decoder (Eq. 6.3) to rely on the localized image regions $\hat{\boldsymbol{r}}_t^l$ to reconstruct the ground-truth sentence, where the localization is conditioned *only* on the generated word from the decoding stage. Our cyclical method can therefore be done without using any annotations of the grounding itself.

Specifically, let $\boldsymbol{y}_t^d = \mathcal{D}^d(\hat{\boldsymbol{r}}_t; \theta_d)$ be the initial language decoder with parameters θ_d (Eq. 6.3), trained to sequentially generate words $\boldsymbol{y}_t^d$. Let $\mathcal{G}(\boldsymbol{y}_t^d; \theta_g)$ define a *localizer* unit with parameters θ_g, that learns to map (ground) each generated word to region(s) in the image, *i.e.,* $\hat{\boldsymbol{r}}_t^l = \mathcal{G}(\boldsymbol{y}_t^d, \boldsymbol{R}; \theta_g)$. Finally, let $\boldsymbol{y}_t^l = \mathcal{D}^l(\hat{\boldsymbol{r}}_t^l; \theta_l)$ be a second decoder, that is required to reconstruct the ground-truth caption using the localized region(s), instead of the attention computed by the decoder itself. We define the cycle:

$$\boldsymbol{y}_t^l = \mathcal{D}^r(\mathcal{G}(\mathcal{D}^d(\hat{\boldsymbol{r}}_t; \theta_d), \boldsymbol{R}; \theta_g); \theta_l), \quad \theta_d = \theta_l, \tag{6.4}$$

where $\mathcal{D}^d$ and $\mathcal{D}^l$ share parameters. Although parameters are shared, the inputs for the two language decoders differ, leading to unique LSTM hidden state values during a run. Note that the Attention LSTMs and logit layers in the two stages also share parameters, though they are omitted for clarity.

Through cyclical joint training, both $\mathcal{D}^d$ and $\mathcal{D}^l$ are required to generate the same ground-truth sentence. They are both optimized to maximize the likelihood of the correct caption:

$$\theta^* = \arg\max_{\theta_d} \sum \log p(\boldsymbol{y}_t^d; \theta_d) + \arg\max_{\theta_l} \sum \log p(\boldsymbol{y}_t^l; \theta_l), \tag{6.5}$$

During training, the localizer regularizes the region attention of the reconstructor and the effect is further propagated to the baseline network in the decoding stage, since the parameters of Attention LSTM and Language LSTM are shared for both decoding and reconstruction stages. Note that the gradient from reconstruction loss will not backprop to the decoder $\mathcal{D}^d$ in the decoding stage since the generated words used as input to the localizer are leafs in the computational graph. The network is implicitly regularized to update its attention mechanism to match with the localized image regions $\hat{\boldsymbol{r}}_t \mapsto \hat{\boldsymbol{r}}_t^l$. In Sec. 6.3.2, we demonstrate that the localized image regions $\hat{\boldsymbol{r}}_t^l$ indeed have higher attention accuracy than $\hat{\boldsymbol{r}}_t$ when using ground-truth words as inputs for the localizer.

6.1.3 Cyclical Training

We now describe each stage of our cyclical model in detail, as illustrated in Figure 6.3.

Decoding. We first use the baseline model presented in Sec. 6.1.1 to generate a sequence of words $\boldsymbol{y} = [\boldsymbol{y}_1^d, \boldsymbol{y}_2^d, ..., \boldsymbol{y}_T^d]$, where T is the ground-truth sentence length.

Localization. Following the decoding process, a localizer $\mathcal{G}$ is then learned to localize the image regions from each generated word $\boldsymbol{y}_t$.

$$\boldsymbol{e}_t = \boldsymbol{W}_e \boldsymbol{y}_t^d, \quad z_{t,n}^l = (\boldsymbol{W}_l \boldsymbol{e}_t)^\top \boldsymbol{r}_n \quad \text{and} \quad \boldsymbol{\beta}_t = \text{softmax}(\boldsymbol{z}_t^l), \tag{6.6}$$

where $\boldsymbol{e}_t$ is the embedding for the word generated during decoding stage at step t, $\boldsymbol{r}_n$ is the image representation of a region proposal, and $\boldsymbol{W}_e$ and $\boldsymbol{W}_l$ are the learned parameters. Based on the localized weights $\boldsymbol{\beta}_t$, the localized region representation can be obtained by $\hat{\boldsymbol{r}}_t^l = \boldsymbol{R}\boldsymbol{\beta}$.

Reconstruction. Finally, the shared language decoder $\mathcal{D}^l$ relies on the localized region representation $\hat{\boldsymbol{r}}_t^l$ to generate the next word. The probability over possible output words is:

$$\boldsymbol{h}_t^L = LSTM_{Lang}([\hat{\boldsymbol{r}}_t^l; \boldsymbol{h}_t^A]), \quad p(\boldsymbol{y}_t^l | \boldsymbol{y}_{1:t-1}^l) = \text{softmax}(\boldsymbol{W}_o \boldsymbol{h}_t^L), \tag{6.7}$$

Given the target ground truth caption $\boldsymbol{y}_{1:T}^*$ and our proposed captioning model parameterized with θ, we minimize the following cross-entropy losses:

$$\mathcal{L}_{CE}(\theta) = -\lambda_1 \underbrace{\sum_{t=1}^{T} log(p_\theta(\boldsymbol{y}_t^* | \boldsymbol{y}_{1:t-1}^*)) \mathbb{1}_{(\boldsymbol{y}_t^* = \boldsymbol{y}_t^d)}}_{\text{decoding loss}} - \lambda_2 \underbrace{\sum_{t=1}^{T} log(p_\theta(\boldsymbol{y}_t^* | \boldsymbol{y}_{1:t-1}^*)) \mathbb{1}_{(\boldsymbol{y}_t^* = \boldsymbol{y}_t^l)}}_{\text{reconstruction loss}} \tag{6.8}$$

where λ_1 and λ_2 are weighting coefficient selected on the validation split.

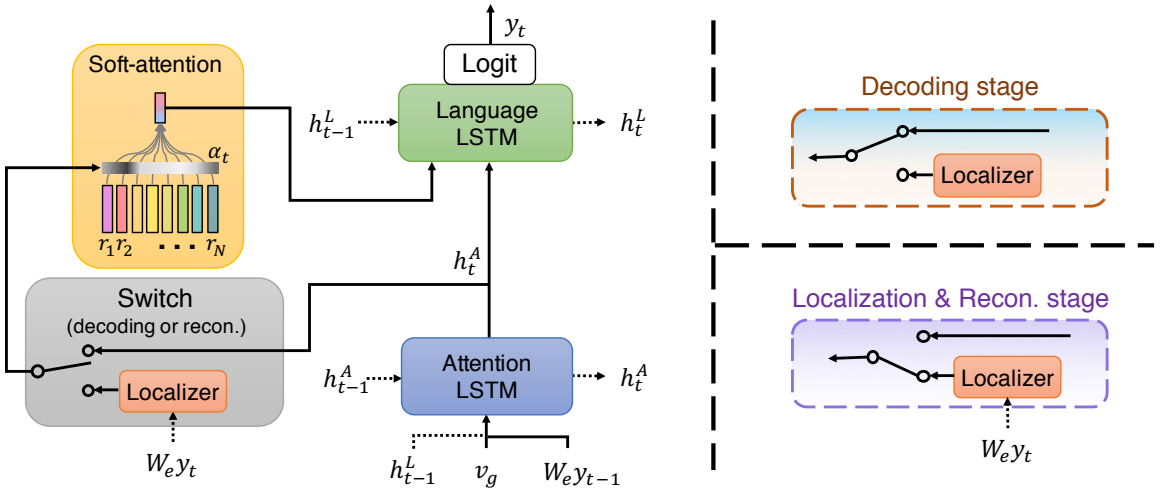

Figure 6.3: Proposed model architecture (left) and how the model operates during decoding, localization, and reconstruction stages (right). During the decoding stage, the soft-attention module uses the hidden state of the Attention LSTM to compute attention weights on image regions. During the localization and reconstruction stage, the soft-attention module instead uses the generated word from decoding stage to compute attention weights on image regions.

6.2 Datasets and Implementations

6.2.1 Datasets

We use the Flickr30k Entities dataset [152] and the ActivityNet-Entities dataset [109] for evaluating our proposed approach. Flickr30k Entities contains 275k annotated bounding boxes from 31k images associated with natural language phrases. Each image is annotated with 5 crowdsourced captions. ActivityNet-Entities contains 15k videos with 158k spatially annotated bounding boxes from 52k video segments.

6.2.2 Evaluation Metrics

Captioning evaluation metrics. We measure captioning performance using four language metrics, including BLEU [147], METEOR [149], CIDEr [150], and SPICE [153].

Grounding evaluation metrics. Following the grounding evaluation from GVD [109],

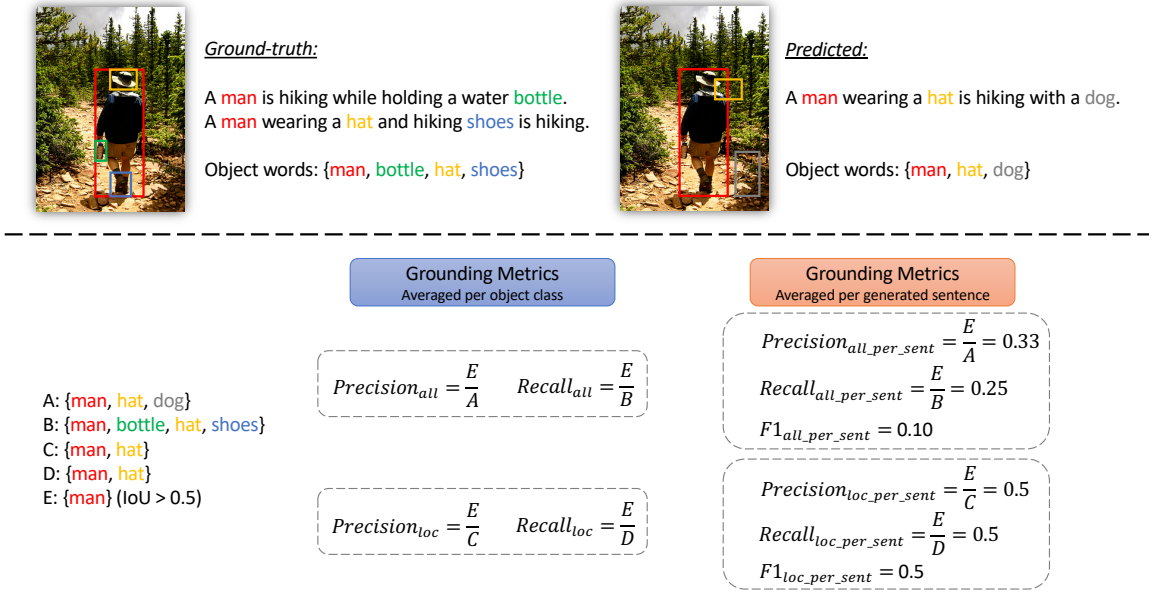

Figure 6.4: Illustration of Grounding metrics.

we measure the attention accuracy on generated sentences, denoted by $F1_{all}$ and $F1_{loc}$. In $F1_{all}$, a region prediction is considered correct if the object word[1] is correctly predicted and also correctly localized. We also compute $F1_{loc}$, which only considers correctly-predicted object words. Please see illustration of the grounding metrics in Fig. 6.4.

In the original formulation, the precision and recall for the two F1 metrics are computed **for each object class**, and it is set to zero if an object class has never been predicted. The scores are computed for each object class and averaged by the total number of classes. Such metrics are extremely stringent as captioning models are generally biased toward certain words in the vocabulary, given the long-tailed distribution of words. In fact, both the baseline and proposed method generate about 45% of the annotated object words within the val set in Flickr30k Entities. The grounding accuracy of the other 55% of the classes are therefore zero, making the averaged grounding accuracy seemingly low.

Measuring grounding per generated sentence. Instead of evaluating grounding on each object class (which might be less intuitive), we include a new grounding evaluation

[1]The object words are words in the sentences that are annotated with corresponding image regions.

metric *per sentence* to directly reflect the grounding measurement of each generated sentence. The metrics are computed against a pool of object words and their ground-truth bounding boxes (GT bbox) collected across five GT captions on Flickr30k Entities (and one GT caption on ActivityNet-Entities). We use the same Prec_{all}, Rec_{all}, Prec_{loc}, and Rec_{loc} as defined previously, but their scores are averaged on each of the generated sentence. As a result, the $\text{F1}_{\text{loc\_per\_sent}}$ measures the F1 score only on the generated words. The model will not be punished if some object words are not generated, but it also needs to maintain diversity to achieve high captioning performance.

6.2.3 Implementation Details

Region proposal features. We use a Faster-RCNN model [102] pre-trained on Visual Genome [154] for region proposal and feature extraction. In practice, besides the region proposal features, we also use the Conv features (*conv4*) extracted from an ImageNet pre-trained ResNet-101. Following GVD [109], the region proposals are represented using the *grounding-aware region encoding*, which is the concatenation of i) region feature, ii) region-class similarity matrix, and iii) location embedding.

For region-class similarity matrix, we define a set of object classifiers as $\boldsymbol{W}_c$, and the region-class similarity matrix can be computed as $M_s = \text{softmax}(W_c^\top \boldsymbol{R})$, which captures the similarity between regions and object classes. We omit the ReLU and Dropout layer after the linear embedding layer for clarity. We initialize $\boldsymbol{W}_c$ using the weight from the last linear layer of an object classifiers pre-trained on the Visual Genome dataset [154].

For location embedding, we use 4 values for the normalized spatial location. The 4-D feature is then projected to a $d_s = 300$-D location embedding for all the regions.

6.2.4 Network architecture.

The embedding dimension for encoding the sentences is 512. We use a dropout layer with ratio 0.5 after the embedding layer. The hidden state size of the Attention and Language

LSTM are 1024. The dimension of other learnable matrices are: $\boldsymbol{W}_e \in \mathbb{R}^{d_v \times 512}$, $\boldsymbol{W}_a \in \mathbb{R}^{1024 \times 512}$, $\boldsymbol{W}_{aa} \in \mathbb{R}^{512 \times 1}$, $\boldsymbol{W}_o \in \mathbb{R}^{1024 \times d_v}$, $\boldsymbol{W}_l \in \mathbb{R}^{512 \times 512}$, where the vocabulary size d_v is 8639 for Flickr30k Entities and 4905 for ActivityNet-Entities.

6.2.5 Training Details

We train the model with ADAM optimizer [155]. The initial learning rate is set to $1e-4$. Learning rates automatically drop by 10x when the CIDEr score is saturated. The batch size is 32 for Flickr30k Entities and 96 for ActivityNet-Entities. We learn the word embedding layer from scratch for fair comparisons with existing work [109]. The hyper-parameters λ_1 and λ_2 are set to 0.5 after hyper-parameter search between 0 and 1.

Flickr30k Entities. In Flickr30k Entities, Images are randomly cropped to 512×512 during training, and resized to 512×512 during inference. Before entering the proposed cyclical training regimen, the decoder was pre-trained for about 35 epochs. The total training epoch with the cyclical training regimen is around 80 epochs. The total training time takes about 1 day.

ActivityNet-Entities. Before entering the proposed cyclical training regimen, the decoder was pre-trained for about 50 epochs. The total training epoch with the cyclical training regimen is around 75 epochs. The total training time takes about 1 day.

6.3 Experiments

6.3.1 Captioning and Grounding Performance Comparison

Flickr30k Entities. We first compare the proposed method with our baseline with or without grounding supervision on the Flickr30k Entities test set (see Table 6.1). To train the supervised baseline, we train the attention mechanism as well as add the region classification task using the ground-truth grounding annotation, similar to GVD [109]. We train the proposed baselines and our method on the training set and choose the best performing checkpoints based on their CIDEr score on the val set. Our experimental results are re-

Table 6.1: Performance comparison on the Flickr30k Entities **test set**: ATT-FCN [97], NBT [87], Up-Down [86], GVD [109], and Baseline is our reimplementation of GVD. *: our results are averaged **across five runs**. Only numbers reported by multiple runs are considered to be bolded.

	Grounding	Captioning Evaluation					Grounding Evaluation			
Method	supervision	B@1	B@4	M	C	S	$F1_{all}$	$F1_{loc}$	$F1_{all\_per\_sent}$	$F1_{loc\_per\_sent}$
ATT-FCN		64.7	19.9	18.5	-	-	-	-	-	-
NBT		69.0	27.1	21.7	57.5	15.6	-	-	-	-
Up-Down		69.4	27.3	21.7	56.6	16.0	4.14	12.3	-	-
GVD (w/o SelfAttn)		69.2	26.9	22.1	60.1	16.1	3.97	11.6	-	-
GVD	✓	69.9	27.3	22.5	62.3	16.5	7.77	22.2	-	-
Baseline*	✓	69.0	26.8	22.4	61.1	16.8	8.44 (+100%)	22.78 (+100%)	27.37 (+100%)	63.19 (+100%)
Baseline*		69.1	26.0	22.1	59.6	16.3	4.08 (+0%)	11.83 (+0%)	13.20 (+0%)	31.83 (+0%)
Cyclical*		**69.4**	**26.9**	**22.3**	**60.8**	**16.6**	**5.11** (+24%)	**14.15** (+21%)	**15.15** (+14%)	**35.56** (+12%)

Table 6.2: Performance comparison on the ActivityNet-Entities **val set**: GVD [109] and Baseline is our reimplementation of GVD. *: our results are averaged **across five runs**. Only numbers reported by multiple runs are considered to be bolded.

	Grounding	Captioning Evaluation					Grounding Evaluation			
Method	supervision	B@1	B@4	M	C	S	$F1_{all}$	$F1_{loc}$	$F1_{all\_per\_sent}$	$F1_{loc\_per\_sent}$
GVD		23.0	2.27	10.7	44.6	13.8	0.28	1.13	-	-
GVD (w/o SelfAttn)		23.2	2.28	10.9	45.6	15.0	3.70	12.7	-	-
GVD	✓	23.9	2.59	11.2	47.5	15.1	7.11	24.1	-	-
Baseline*	✓	23.1	2.13	10.7	45.0	14.6	7.30 (+100%)	25.02 (+100%)	17.88 (+100%)	60.23 (+100%)
Baseline*		23.2	2.22	10.8	45.9	**15.1**	3.75 (+0%)	12.00 (+0%)	9.41 (+0%)	31.68 (+0%)
Cyclical*		**23.7**	**2.45**	**11.1**	**46.4**	14.8	**4.68** (+26%)	**15.84** (+29%)	**12.60** (+38%)	**44.04** (+43%)

ported by averaging across five runs on the test set. We report only the mean of the five runs to keep the table uncluttered. When compared to the existing state of the arts, our proposed baselines achieve comparable captioning evaluation performances and grounding accuracy. Using the resulting supervised baseline as the upper bound, our proposed method with cyclical training statistically achieves around 20 to 25% relative grounding accuracy improvements for both $F1_{\text{all}}$ and $F1_{\text{loc}}$ and 10 to 15% for $F1_{\text{all\_per\_sent}}$ and $F1_{\text{loc\_per\_sent}}$ without utilizing any grounding annotations or additional computation during inference.

ActivityNet-Entities. We adapt our proposed baselines and method to the ActivityNet-Entities video dataset (see Table 6.2). We can see that our proposed method significantly improved the grounding accuracy around 25% to 30% relative grounding accuracy improvements for both $F1_{\text{all}}$ and $F1_{\text{loc}}$ and around 40% for $F1_{\text{all\_per\_sent}}$ and $F1_{\text{loc\_per\_sent}}$.

Table 6.3: Grounding performance when using better object detector on the Flickr30k Entities **test** set (results are averaged three runs). Fully-supervised method (Sup.) is used as upper bound, thus its numbers are not bolded.

Method	Grounding supervision	Captioning Evaluation					Grounding Evaluation			
		B@1	B@4	M	C	S	$F1_{all}$	$F1_{loc}$	$F1_{all\_per\_sent}$	$F1_{loc\_per\_sent}$
Unrealistically perfect object detector										
Baseline	✓	75.6	32.0	25.3	75.6	22.3	23.19 (+100%)	52.83 (+100%)	51.43 (+100%)	90.76 (+100%)
Baseline		75.1	32.1	25.2	76.3	22.0	20.82 (+0%)	48.74 (+0%)	43.21 (+0%)	77.81 (+0%)
Cyclical		**76.7**	**32.8**	**25.8**	**80.2**	**22.7**	**25.27** (+188%)	**54.54** (+142%)	**46.98** (+46%)	**81.56** (+29%)
Grounding-biased object detector										
Baseline	✓	65.9	23.4	21.3	53.3	15.5	8.23 (+100%)	23.95 (+100%)	28.06 (+100%)	66.96 (+100%)
Baseline		**66.1**	**23.5**	**21.2**	**52.4**	**15.4**	5.95 (+0%)	17.51 (+0%)	18.11 (+0%)	42.84 (+0%)
Cyclical		65.5	23.3	**21.2**	52.0	**15.4**	**6.87** (+40%)	**19.65** (+33%)	**20.82** (+27%)	**50.25** (+31%)

6.3.2 Analysis

Are localized image regions better than attended image regions during training? Considering our intuition described in Sec. 6.1, we expect the decoder to be regularized to update its attention mechanism to match with the localized image regions $\hat{\boldsymbol{r}}_t \mapsto \hat{\boldsymbol{r}}^l_t$. This indicates that the localized image regions should be more accurate than the attended image regions by the decoder during training. To verify this, we compute the attention accuracy for both decoder and localizer over ground-truth sentences following [105, 156]. The attention accuracy for localizer is 20.4% and is higher than the 19.3% from the decoder at the end of training, which confirms our hypothesis.

Grounding performance when using a *better* object detector. In Table 6.1 and 6.2 we showed that our proposed method significantly improved the grounding accuracy for both image and video captioning. These experimental settings follow the widely used procedure for visual captioning systems: extract regional proposal features and generate visual captions by attending to those extracted visual features. One might ask, what if we have a better object detector that can extract robust visual representation that are better aligned with the word embeddings? Will visual grounding still an issue for captioning?

To answer this, we ran two sets of experiments (Table 6.3): **(1) *Perfect* object detector**: we replace the ROIs by ground-truth bbox and represent the new ROIs by learning embedding features directly from ground-truth object words associated with each ground-truth

bbox. This experiment gives an estimate of the captioning and grounding performance if we have (almost) perfect ROI representations (though unrealistic). We can see that the fully-supervised method achieves an $F1_{all}$ of only 23%, which further confirms the difficulty of the metric and the necessity of our grounding metric on a per sentence level (note that $F1_{loc\_per\_sent}$ shows 90%). We can also see that baseline (unsup.) still leaves room for improvement on grounding performance. Surprisingly, our method improved both captioning and grounding accuracy and surpasses the fully-supervised baseline except on the $F1_{loc\_per\_sent}$. We find that it is because the baseline (sup.) overfits to the training set, while ours is regularized from the cyclical training. Also, our generated object words are more diverse, which is critical for $F1_{all}$ and $F1_{loc}$. **(2) *Grounding-biased* object detector**: we extract ROI features from an object detector pre-trained on Flickr30k. Thus, the ROI features and their associated object predictions are biased toward the annotated object words but do not generalize to predict diverse captions compared to the original object detector trained from Visual Genome, resulting in lower captioning performance. We can see that our proposed method still successfully improves grounding and maintains captioning performance in this experiment setting as well.

How does the number of annotations affect grounding performance? In Figure 6.5, we present the average F1-score on the Flickr30k Entities val set when grouping classes according to their frequency of appearance in the training set[3]. We see that, unsurprisingly, the largest difference in grounding accuracy between the supervised and our proposed cyclical training is for the 50 most frequently appearing object classes, where enough training data exists. As the number of annotated boxes decreases, however, the difference in performance diminishes, and cyclical training appears to be more robust. Overall, we see that the supervised method is biased towards frequently appearing objects, while grounding performance for the proposed approach is more balanced among classes.

Should we explicitly make attended image regions to be similar to localized image

[3]We group the 460 object classes in 10 groups, sorted by the number of annotated bounding boxes.

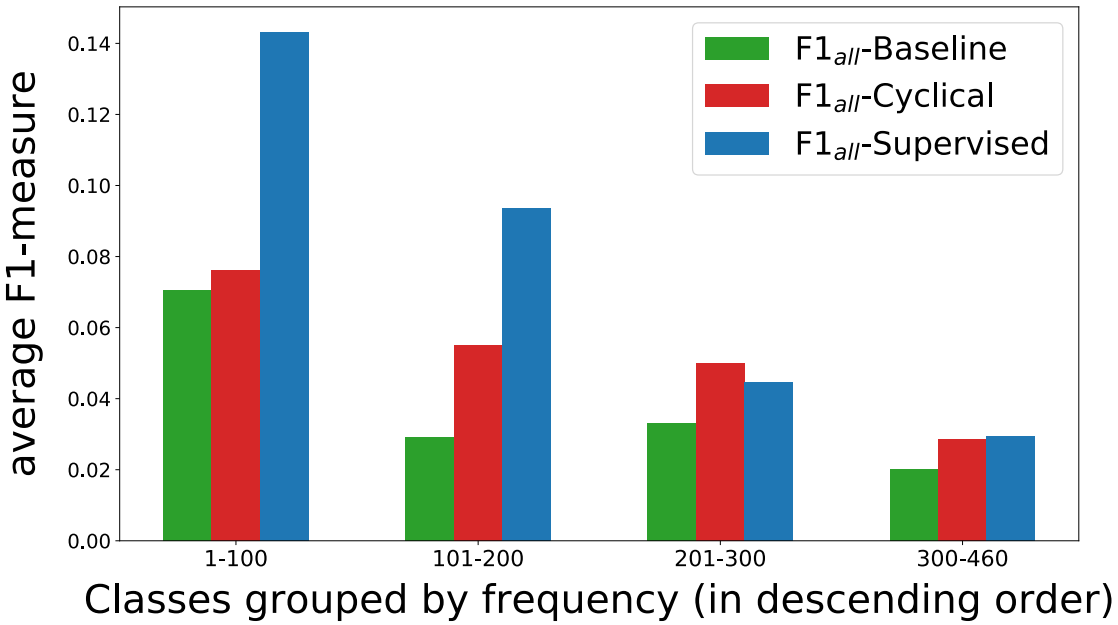

Figure 6.5: Average F1$_{all}$-score per class as a function of class frequency.

regions? One possible way to regularize the attention mechanism of the decoder is to explicitly optimize $\hat{\boldsymbol{r}}_t \mapsto \hat{\boldsymbol{r}}_t^l$ via KL divergence over two soft-attention weights $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$. The experimental results are shown in Table 6.4 (*Attention consistency*). We use a single run unsupervised baseline with a fix random seed as baseline model for ablation study. We can see that when explicitly forcing the attended regions to be similar to the localized regions, both the captioning performance and the grounding accuracy remain similar to the baseline (unsup.). We conjecture that this is due to the noisy localized regions at the initial training stage. When forcing the attended regions to be similar to noisy localized regions, the Language LSTM will eventually learn to not rely on the attended region at each step for generating sequence of words. To verify, we increase the weight for attention consistency loss and observed that it has lower grounding accuracy (F1$_{\text{all}}$ = 3.2), but the captioning will reach similar performance while taking 1.5x longer to reach convergence.

Is using only the generated word for localization necessarily? Our proposed local-

Table 6.4: Model ablation study on the Flickr30k Entities val set.

	Captioning Eval.			Grounding Eval.	
#	M	C	S	$F1_{all}$	$F1_{loc}$
Baseline (Unsup.)	**22.3**	62.1	16.0	4.18	11.9
Cyclical	22.2	**62.2**	**16.2**	**5.63**	**14.6**
- Attention consistency	**22.3**	61.8	**16.2**	4.19	11.3
- Localizer using $\boldsymbol{h}^A$	22.2	61.8	16.1	4.58	11.3

Table 6.5: Performance comparison on the Flickr30k Entities **test set**. All results are averaged **across five runs**.

	Captioning Evaluation					Grounding Evaluation			
Method	B@1	B@4	M	C	S	$F1_{all}$	$F1_{loc}$	$F1_{all\_per\_sent}$	$F1_{loc\_per\_sent}$
Baseline	69.1	26.0	22.1	59.6	16.3	4.08	11.83	13.20	31.83
Cyclical	69.4	26.9	22.3	60.8	**16.6**	5.11	14.15	15.15	35.56
Cyclical (1)	69.7	27.0	22.2	60.1	16.5	**5.14**	**14.32**	15.36	36.33
Cyclical (2)	**69.9**	**27.5**	**22.4**	**62.0**	**16.6**	5.13	13.99	**16.30**	**38.45**

izer (Eq. 6.6 and Figure 6.3) relies on purely the word embedding representation to locate the image regions. This forces the localizer to rely only on the word embedding without biasing it with the memorized information from the Attention LSTM. As shown in the Table 6.4 (localizer using $\boldsymbol{h}^A$), although this achieves comparable captioning performance, it has lower grounding accuracy improvement compared to our proposed method.

Can words that are not visually-groundable handled differently? In the proposed approach, all the words are handled the same regardless of whether they are visually-groundable or not. Yet, typically words that are nouns or verbs are more likely to be grounded, and words like ”a”, ”the”, *etc*, are not visually-groundable.

We explored a few method variants to handle nouns and verbs differently. Mainly, we explored with two variants. Cyclical (1): the reconstruction loss is only computed when the target word is either nouns or verbs. Cyclical (2): the localized region representation will be invalid (set to zero) if the target word is neither nouns nor verbs.

The experimental results are shown in Table 6.5, 6.6, and 6.7. For the first variant, Cyclical (1), we observed that the captioning performance stays the same while grounding

Table 6.6: Performance comparison on the ActivityNet-Entities **val set**. All results are averaged **across five runs**.

	Captioning Evaluation					Grounding Evaluation			
Method	B@1	B@4	M	C	S	$F1_{all}$	$F1_{loc}$	$F1_{all\_per\_sent}$	$F1_{loc\_per\_sent}$
Baseline	23.2	2.22	10.8	45.9	**15.1**	3.75	12.00	9.41	31.68
Cyclical	23.7	2.45	11.1	46.4	14.8	**4.68**	**15.84**	**12.60**	**44.04**
Cyclical (2)	**23.9**	**2.58**	**11.2**	**46.6**	14.8	4.48	15.01	11.53	40.30

Table 6.7: Grounding performance when using better object detector on the Flickr30k Entities **test** set (results are averaged three runs).

	Captioning Evaluation					Grounding Evaluation			
Method	B@1	B@4	M	C	S	$F1_{all}$	$F1_{loc}$	$F1_{all\_per\_sent}$	$F1_{loc\_per\_sent}$
Unrealistically perfect object detector									
Baseline	75.1	32.1	25.2	76.3	22.0	20.82	48.74	43.21	77.81
Cyclical	**76.7**	**32.8**	**25.8**	**80.2**	**22.7**	25.27	54.54	46.98	81.56
Cyclical (2)	75.8	32.2	25.6	79.0	22.4	**25.65**	**55.81**	**48.99**	**85.99**

accuracy has a small improvement. On the other hand, for the second variant, Cyclical (2), we can see that all captioning scores are improved over baseline with CIDEr improved 2.4. We can also see that grounding accuracy on per sentence basis further improved as well. We then conducted further experiments on both ActivityNet-Entities and Flickr30k Entities with *unrealistically perfect object detector*, but the improvements however are not consistent. In summary: on the Flickr30k Entities test set, we observed that CIDEr is better and grounding per sentence better, on the ActivityNet-Entities val set, the captioning performances are about the same but grounding accuracy became worse, and on the Flickr30k Entities test set with unrealistically perfect object detector, captioning performances are slightly worse but grounding accuracy improved.

6.4 Human Evaluation on grounding

We conduct a human evaluation on the perceptual quality of the grounding. We asked 10 human subjects to pick the best among two grounded regions (by baseline and Cyclical) for each word. The subjects have three options to choose from: 1) grounded region A is

better, 2) grounded region B is better, and 3) they are about the same (see Figure 6.6 for example). Authors or other colleagues familiar with the proposed method were excluded from the study. Each of the human subjects were given 25 images, each with a varying number of groundable words. Each image was presented to two different human subjects in order to be able to measure inter-rater agreement. To avoid being biased towards the "object words" defined in the dataset for automatic grounding evaluation, for the study we define a word to be groundable if it is either a noun or verb. The order of approaches was randomized for each sentence.

Our experiment on the Flickr30k Entities val set showed that: 28.1% of words are more grounded by Cyclical, 24.8% of words are more grounded by baseline, and 47.1% of words are similarly grounded.

We also measured inter-rater agreement between each pair of human subjects: 72.7% of ratings are the same, 4.9% of ratings are the opposite, and 22.4% of ratings could be ambiguous (*e.g.,* one chose A is better, the other chose they are about the same).

We would also like to make a note that the grounded words judged to be similar largely consisted of very easy or impossible cases. For example, words like *mountain, water, street, etc,* are typically rated to be "about the same" since they usually have many possible boxes and is very easy for both models to ground the words correctly. On the other hand, for visually ungroundable cases, *e.g., stand* appears a lot and the subject would choose *about the same* since the image does not cover the fact that the person's feet are on the ground.

We see that the human study results follow the grounding results presented in the paper and show an improvement in grounding accuracy for the proposed method over a strong baseline. The improvement is achieved without grounding annotations or extra computation at test time.

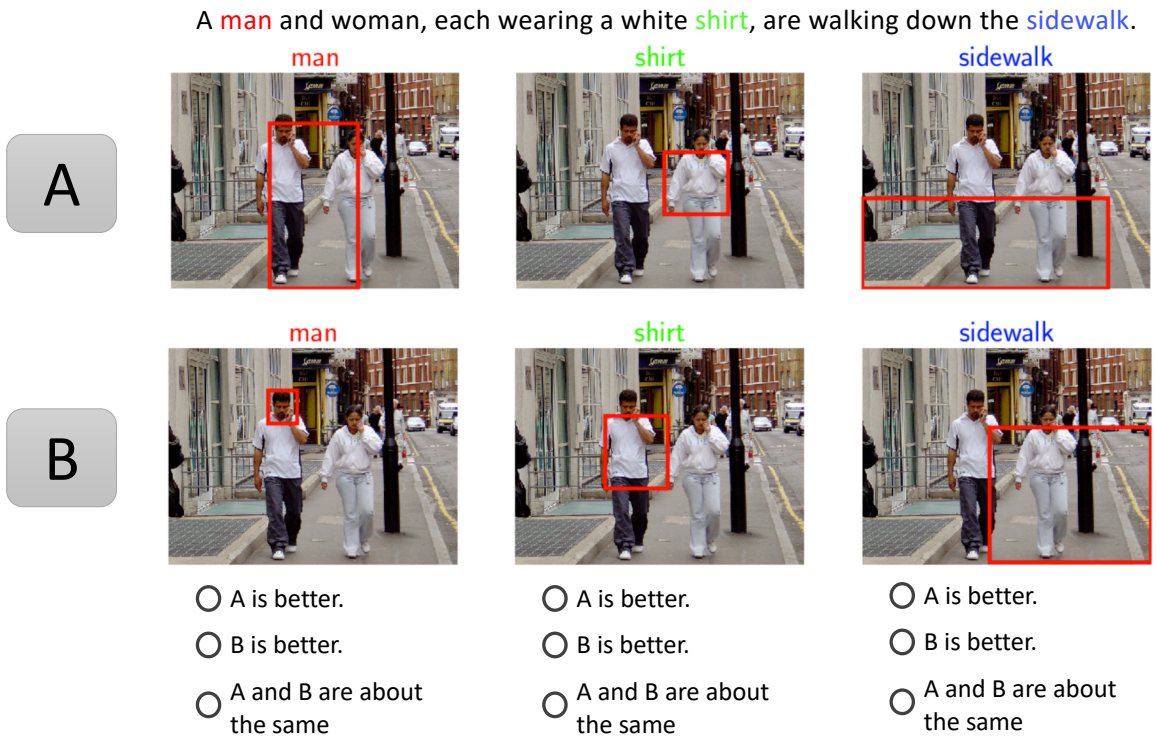

Figure 6.6: Demonstration of our human evaluation study on grounding. Each human subject is required to rate which method (A or B) has a better grounding on each highlighted word.

6.5 Qualitative Analysis

We additionally conduct qualitative analysis for comparing the baseline (Unsup.) and the proposed method in Figure 6.8. Each highlighted word has a corresponding image region annotated on the original image. The image regions are selected based on the region with the maximum attention weight in α_t. We can see that our proposed method significantly outperformed the baseline (Unsup.) in terms of both the quality of the generated sentence and grounding accuracy.

In Figure 6.9, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, and 6.16, we illustrated the sequence of attended image region when generating each word for a complete image description. At each step, only the top-1 attended image region is shown. This is the same as how the grounding accuracy is measured. Please see the description for Figure 6.9 - 6.16 for further

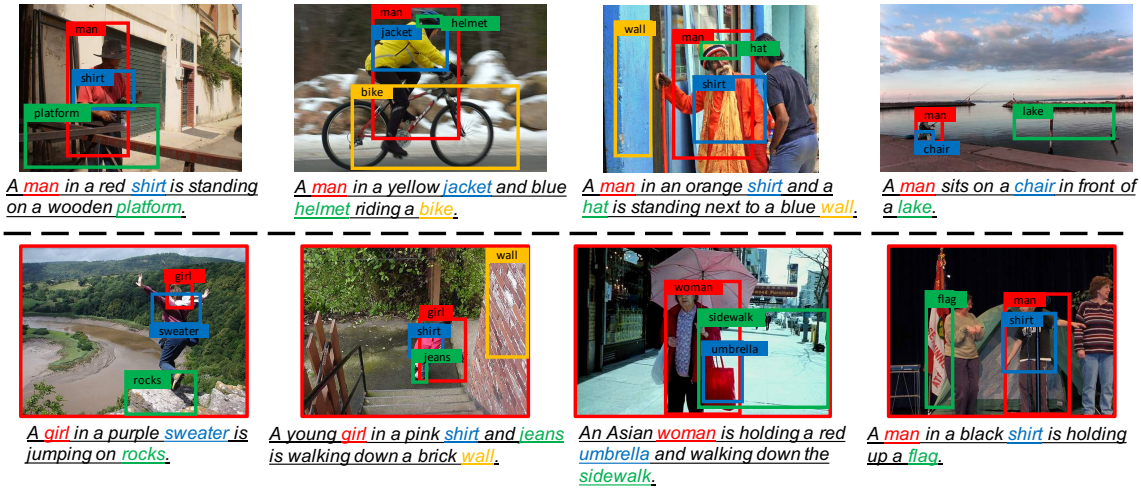

Figure 6.7: Correct (top) examples and examples with errors (bottom) from the proposed method.

discussions on the qualitative results.

In addition, we also discuss a number of correct and incorrect examples of our proposed method in Figure 6.7. We observe that while the model is able to generate grounded captions for the images, it may sometimes overlook the semantic meaning of the generated sentences, for example, *"A young girl [...] walking down a brick wall"*. Similarly, the model can overlook the spatial relationship between the objects, for instance, *"A man [...] is holding up a flag"*.

Figure 6.8: Generated captions and corresponding visual grounding regions with comparison between baseline (left) and proposed approach (right). Our proposed method is able to generate more descriptive sentences while selecting the correct regions for generating the corresponding words.

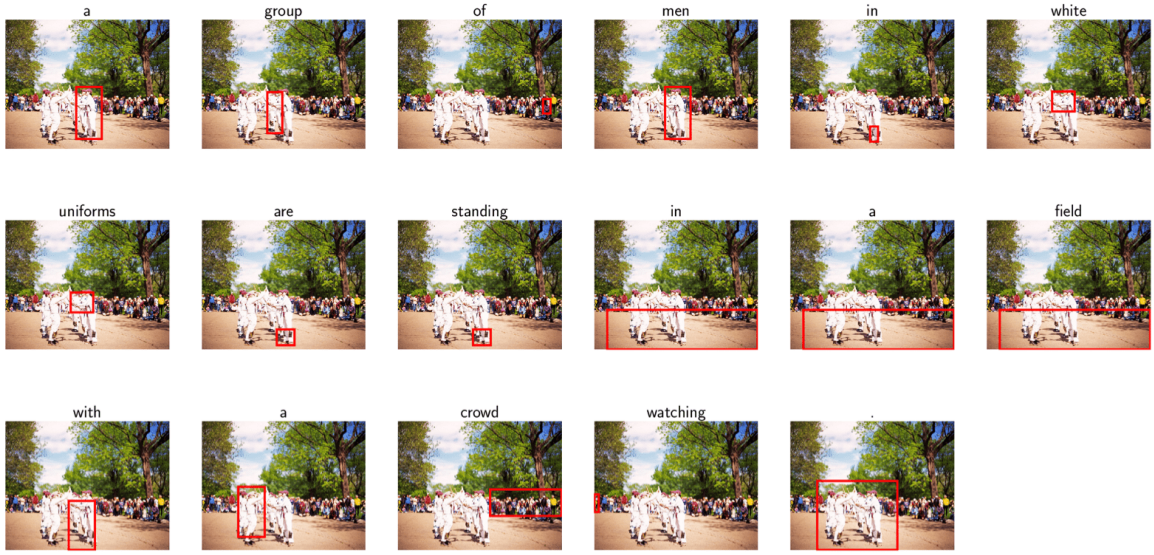

Figure 6.9: *A group of men in white uniforms are standing in a field with a crowd watching.* We can see that our proposed method attends to the sensible image regions for generating visually-groundable words, e.g., *man*, *uniforms*, *field*, and *crowd*. Interestingly, when generating *standing*, the model pays its attention on the image region with a foot on the ground.

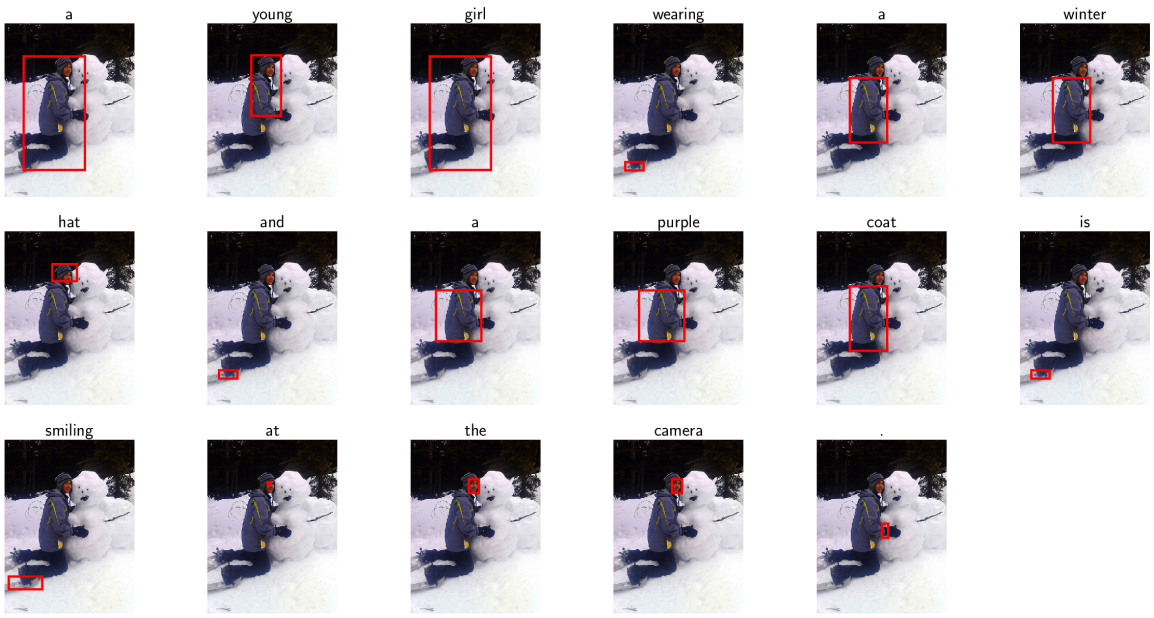

Figure 6.10: *A young girl wearing a winter hat and a purple coat is smiling at the camera.* The proposed method is able to select the corresponding image regions to generate *girl*, *hat*, and *coat* correctly. We have also observed that the model tends to localize the person's face when generating *camera*.

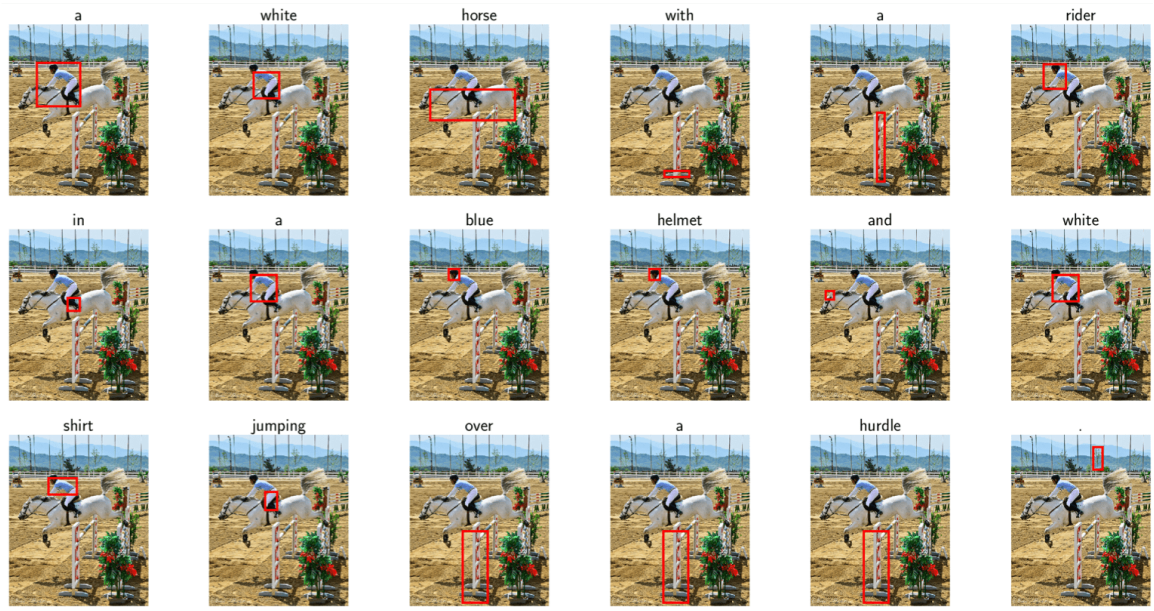

Figure 6.11: *A white horse with a rider in a blue helmet and white shirt jumping over a hurdle.* While the model is able to correctly locate objects such as *horse*, *rider*, *helmet*, *shirt*, and *hurdle*, it mistakenly describes the rider as wearing a blue helmet, while it's actually black, and with white shirt while it's blue.

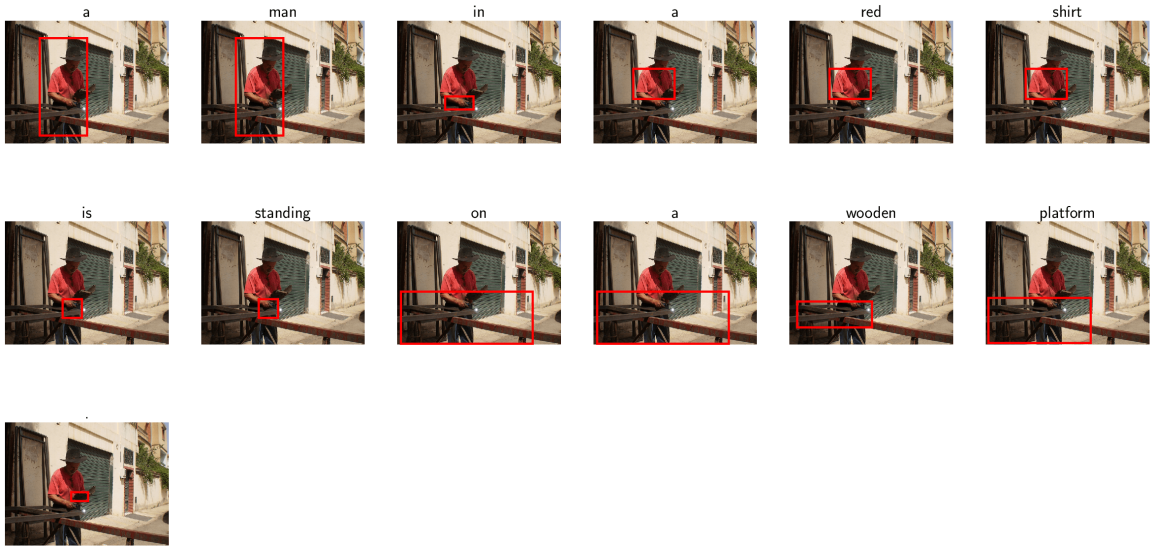

Figure 6.12: *A man in a red shirt is standing on a wooden platform.* Our method correctly attends on the correct regions for generating *man*, *shirt*, and *platform*.

Figure 6.13: *A man in a yellow jacket and blue helmet riding a bike.* The proposed method correctly generates a descriptive sentence while precisely attending to the image regions for each visually-groundable words: *man*, *jacket*, *helmet*, and *bike*.

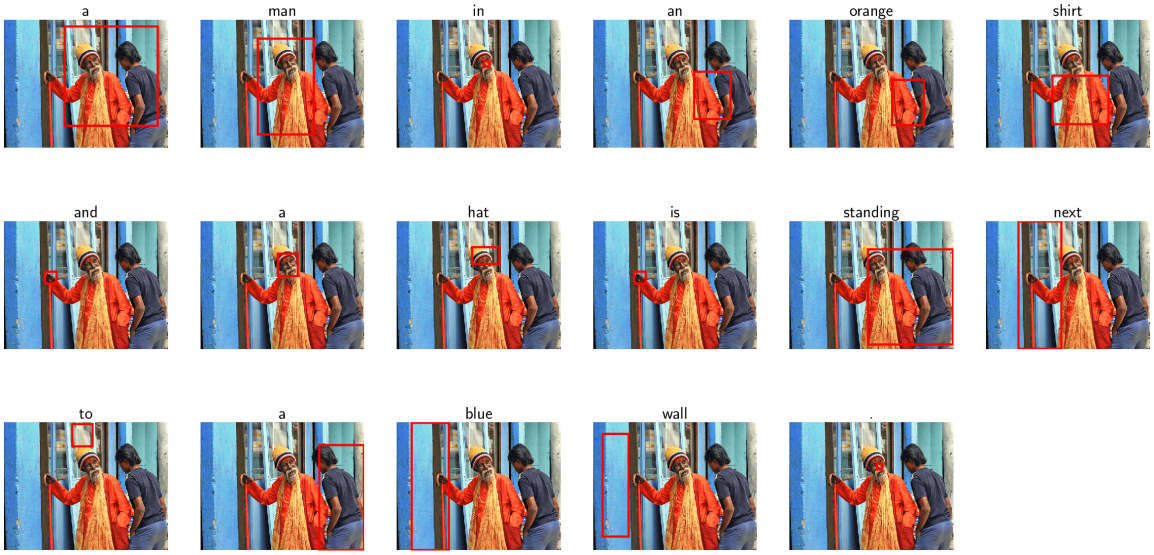

Figure 6.14: *A man in an orange shirt and a hat is standing next to a blue wall.* While our method is able to ground the generated sentence on the objects like: *man*, *shirt*, *hat*, and *wall* , it completely ignores the person standing next to the man in the orange cloth.

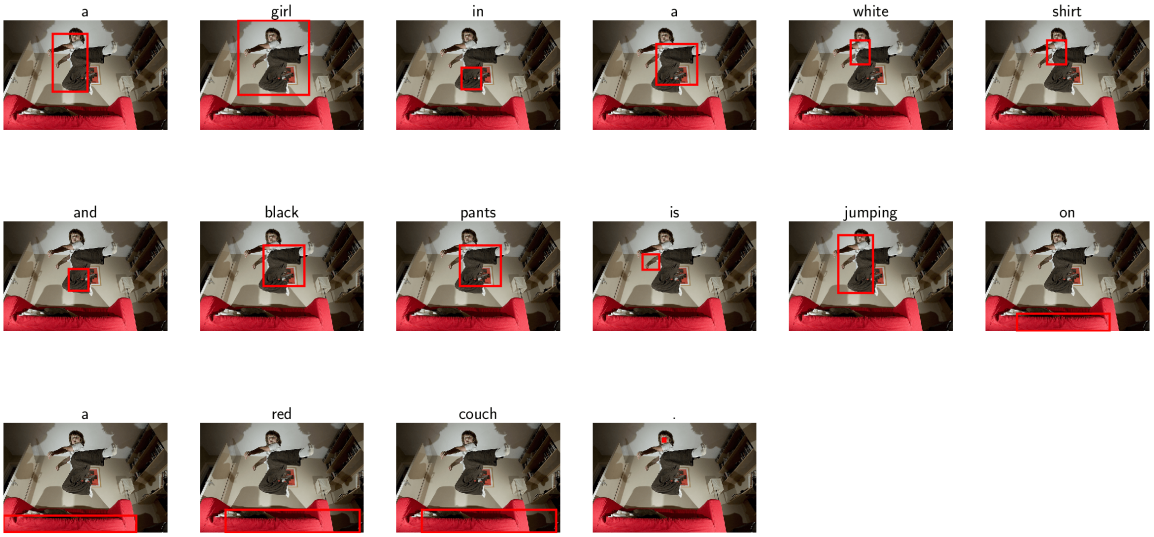

Figure 6.15: *A girl in a white shirt and black pants is jumping on a red couch.* Our method is able to ground the generated descriptive sentence with the correct grounding on: *girl*, *shirt*, *pants*, and *couch*.

Figure 6.16: *A man in a blue robe walks down a cobblestone street.* Our method grounds the visually-relevant words like: *man*, *robe*, and *street*. We can also see that it is able to locate the foot on ground for *walks*.

CHAPTER 7
CONCLUSION

7.1 Contributions

The Vision-and-Language Navigation (VLN) task entails an agent following navigational instruction in photo-realistic unknown environments. Unlike previous static vision and language tasks, this challenging task demands that the agent be aware of which instruction was completed, which instruction is needed next, which way to go, and its navigation progress towards the goal. While existing approaches utilize an attentional mechanism on the instructions, we qualitatively show that the resulting focus may not tend to track which instruction is next nor progress towards the goal. To overcome this issue, an ideal navigation agent should (1) be aware of the current state, *i.e.,* which part of the instruction is completed and what will be next, (2) estimate which navigation direction represented by an image matches with the part of the instruction to be carried out, and (3) ensure the grounded instruction correctly reflects the progress toward the goal. To do so, we introduce a *self-monitoring* agent with two complementary components in Chapter 3: (1) visual-textual co-grounding module to locate the instruction completed in the past, the instruction required for the next action, and the next moving direction from surrounding images and (2) progress monitor to ensure the grounded instruction correctly reflects the navigation progress. We test our self-monitoring agent on a standard benchmark and analyze our proposed approach through a series of ablation studies that elucidate the contributions of the primary components. Using our proposed method, we set the new state of the art by a significant margin (8% absolute increase in success rate on the unseen test set).

In developing the self-monitoring navigation agent, we discussed two scenario for running the inference of the trained navigation agent: with or without beam search. While in

both inference modes, our proposed self-monitoring navigation agent achieve state-of-the-art performances, but relying on beam search, which thoroughly explores a large number of trajectories, is unrealistic for applications such as robotics.

In Chapter 4, inspired by the intuition of viewing the problem as search on a navigation graph, we propose to use a progress monitor developed as a learnable heuristic for search. We then propose two modules incorporated into an end-to-end architecture: 1) A learned mechanism to perform backtracking, which decides whether to continue moving forward or roll back to a previous state (Regret Module) and 2) A mechanism to help the agent decide which direction to go next by showing directions that are visited and their associated progress estimate (Progress Marker). Combined, the proposed approach significantly outperforms state-of-the-art methods using greedy action selection, with 5% absolute improvement on the test server in success rates, and more importantly 8% on success rates normalized by the path length.

In Chapter 3 and 4, we demonstrated how to enforce the navigation agent to be more grounded in both textual and visual information. After improved the grounding in VLN, in Chapter 5, we then discuss how to ground high-level video concepts like human actions or sentence descriptions into regions and their interactions in the spatiotemporal domain. Specifically, we propose to efficiently learn higher-order interactions between arbitrary subgroups of objects for fine-grained video understanding. We demonstrate that modeling object interactions significantly improves accuracy for both action recognition and video captioning, while saving more than 3-times the computation over traditional pairwise relationships. The proposed method is validated on two large-scale datasets: Kinetics and ActivityNet Captions. Our SINet and SINet-Caption achieve state-of-the-art performances on both datasets even though the videos are sampled at a maximum of 1 FPS. To the best of our knowledge, this is the first work modeling object interactions on open domain large-scale video datasets, and we additionally model higher-order object interactions which improves the performance with low computational costs.

In Chapter 5.2, we proposed a captioning model that can leverage the object-level relational reasoning for video understanding. However, when automatically generating a sentence description for an image or video, it often remains unclear how well the generated caption is grounded, or if the model hallucinates based on priors in the dataset and/or the language model. The most common way of relating image regions with words in caption models is through an attention mechanism over the regions that are used as input to predict the next word. The model must therefore learn to predict the attentional weights without knowing the word it should localize. This is difficult to train without grounding supervision since recurrent models can propagate past information and there is no explicit signal to force the captioning model to properly ground the individual decoded words. In Chapter 6, we help the model to achieve this via a novel cyclical training regimen that forces the model to localize each word in the image *after* the sentence decoder generates it, and then reconstruct the sentence from the localized image region(s) to match the ground-truth. The initial language decoder and the proposed reconstructor share parameters during training and are learned jointly with the localizer, allowing the model to regularize the attention mechanism. Our proposed framework only requires learning one extra fully-connected layer (the localizer), a layer that can be removed at test time. We show that our model significantly improves grounding accuracy without relying on grounding supervision or introducing extra computation during inference for both image and video captioning tasks.

7.2 Prospective Research Directions

7.2.1 Vision-and-Language Navigation

In this thesis, we proposed many approaches for problems that required grounding to be improved. In this section, we will discuss some prospective research directions to each of the problem and finally discuss self-supervised representation learning as a research direction that could be universally applicable for all visual-textual grounding problems.

In VLN, our ultimate direction is to have an actual AI agent performing navigation

or object manipulation in the real world. This requires several important research explorations. For example, first, extending VLN setting to various virtual environments and developing methods that can transfer trained agents from one virtual environment to another virtual environment. This transfer could mean: (1) visual transfer, which has the same type of environment like houses with the same navigation task; (2) environmental transfer, which could be transferring an agent trained in *house* environment to *office* environment; (3) task transfer, which transfers learned sub-tasks to perform compositions of the sub-tasks.

On the other hand, it is also essential to test a trained agent by deploying it to the real world. In addition to the challenges like visual transfer or environmental transfer, this also requires that the virtual environment should have the same local motion for the agent as the real-world robot. A number of efforts are indeed paving the road for this research direction, *e.g.,* PyRobot [157], Habitat [158], Gibson [159, 160], etc.

7.2.2 Visual Captioning

To make a visual captioning system applicable to the real work, the trained model needs to be trustworthy. While improving visual grounding accuracy of the captioning model certainly makes it becomes more trustworthy, the relative improvement on the captioning metrics is still small. Yet, it is also known that captioning metrics many times do not reflect the quality of the generated sentence description, and many state-of-the-art methods already outperformed humans on various captioning metrics. To the best of my knowledge, research directions that continue to improve the captioning evaluation metrics seem to be the steps needed shortly, which will further enable us to exam how improving grounding accuracy affects the captioning performance. Lastly, our proposed cyclical training regimen enables us to train the visual captioning model without grounding annotations. This gives us an opportunity to train such a model on much larger datasets, which do not have grounding annotations, like Conceptual Captions dataset with three million images [161],

to further exam how does grounding accuracy changes when transferring captioning model from a larger dataset to our target dataset.

7.2.3 Self-supervised Representation Learning

All tasks and problem sets that I tackled in this thesis involve various usages of visual representations, *e.g.,* visual representations from region proposal network for action recognition and visual captioning models or image representations of navigable directions in virtual environments for navigation agents. Our proposed methods assumes these representations are (mostly) fixed through out the model training. However, these representations can largely affect the downstream task performances, especially on grounding between visual and textual representations. Ideally, we would like the visual representation to be universally generalized. However, currently, they were obtained from networks pre-trained from ImageNet or MS-COCO, which seems to be less ideal.

On the other hand, self-supervised learning methods aim at learning generalized representations regardless of the downstream tasks. These methods learn visual representations without biased to certain downstream tasks, which arguably could learn more generalized visual representations. Besides, the overall goal of this thesis is to leverage information from both vision and language domains. Recently, in the language side, researchers have made significant progress toward learning generalized language representation by using self-supervised learning, *e.g.,* GPT [162], BERT [163], etc. To make progress for domains at the intersection of vision and language, significant progress in self-supervised learning for vision is much needed.

REFERENCES

[1] C.-Y. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong, "Self-monitoring navigation agent via auxiliary progress estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[2] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[3] X. Wang, W. Xiong, H. Wang, and W. Y. Wang, "Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation," in *European Conference on Computer Vision (ECCV)*, 2018.

[4] C.-Y. Ma, M.-H. Chen, Z. Kira, and G. AlRegib, "Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition," *Signal Processing: Image Communication*, 2018.

[5] C.-Y. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. P. Graf, "Attend and interact: Higher-order object interactions for video understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 20–36.

[7] L. Anne Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell, "Localizing moments in video with natural language," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5803–5812.

[8] L. A. Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell, "Localizing moments in video with temporal language.," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[9] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *In Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, 2016, pp. 1–8.

[10] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 3357–3364.

[11] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[12] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "Learning to navigate in cities without a map," *arXiv preprint arXiv:1804.00168*, 2018.

[13] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2018.

[14] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin, "Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification," *arXiv preprint arXiv:1708.03805*, 2017.

[15] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," *arXiv preprint arXiv:1706.06905*, 2017.

[16] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, "Asynchronous temporal fields for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[17] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[18] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[19] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3d: Generic features for video analysis," *CoRR, abs/1412.0767*, vol. 2, p. 7, 2014.

[20] C. Liu, J. Mao, F. Sha, and A. Yuille, "Attention correctness in neural image captioning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[21] L. Anne Hendricks, K. Burns, K. Saenko, T. Darrell, and A. Rohrbach, "Women also snowboard: Overcoming bias in captioning models," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 771–787.

[22] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, "Object hallucination in image captioning," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 4035–4045.

[23] D. Gurari, Q. Li, A. J. Stangl, A. Guo, C. Lin, K. Grauman, J. Luo, and J. P. Bigham, "Vizwiz grand challenge: Answering visual questions from blind people," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3608–3617.

[24] L. Zhou, C. Xu, and J. J. Corso, "Towards automatic learning of procedures from web instructional videos," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[25] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8739–8748.

[26] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2425–2433.

[27] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "Movieqa: Understanding stories in movies through question-answering," in *In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4631–4640.

[28] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra, "Visual Dialog," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] D. K. Misra, J. Sung, K. Lee, and A. Saxena, "Tell me dave: Context-sensitive grounding of natural language to manipulation instructions," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 281–300, 2016.

[30] J. Arkin, M. R. Walter, A. Boteanu, M. E. Napoli, H. Biggie, H. Kress-Gazit, and T. M. Howard, "Contextual awareness: Understanding monologic natural language instructions for autonomous robots," in *Robot and Human Interactive Communication (RO-MAN), 2017 26th IEEE International Symposium on*, IEEE, 2017, pp. 502–509.

[31] M. Al-Omari, P. Duckworth, D. C. Hogg, and A. G. Cohn, "Natural language acquisition and grounding for embodied robotic systems.," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017, pp. 4349–4356.

[32] T. Kollar, J. Krishnamurthy, and G. P. Strimel, "Toward interactive grounded language acqusition.," in *Robotics: Science and systems*, vol. 1, 2013, pp. 721–732.

[33] M. Spranger and L. Steels, "Co-acquisition of syntax and semantics-an investigation in spatial language," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

[34] K. S. Dubba, M. R. De Oliveira, G. H. Lim, H. Kasaei, L. S. Lopes, A. Tomé, and A. G. Cohn, "Grounding language in perception for scene conceptualization in autonomous robots," in *Qualitative Representations for Robots: Papers from the AAAI Spring Symposium. Technical report*, AI Access Foundation, 2014, pp. 26–33.

[35] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[36] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4089–4098.

[37] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *Experimental Robotics*, Springer, 2013, pp. 403–415.

[38] S. Hemachandra, F. Duvallet, T. M. Howard, N. Roy, A. Stentz, and M. R. Walter, "Learning models for following natural language directions in unknown environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, 2015.

[39] F. Duvallet, M. R. Walter, T. Howard, S. Hemachandra, J. Oh, S. Teller, N. Roy, and A. Stentz, "Inferring maps and behaviors from natural language instructions," in *Experimental Robotics*, Springer, 2016, pp. 373–388.

[40] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela, "Talk the walk: Navigating new york city through grounded dialogue," *arXiv preprint arXiv:1807.03367*, 2018.

[41] E. K. D. F. L. F.-F. A. G. R. M. A. F. Yuke Zhu Daniel Gordon, "Visual Semantic Planning using Deep Successor Representations," in *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[42] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson, "Visual representations for semantic target driven navigation," *arXiv preprint arXiv:1805.06066*, 2018.

[43] G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Ahuja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, *et al.*, "Unsupervised predictive memory in a goal-directed agent," *arXiv preprint arXiv:1803.10760*, 2018.

[44] T.-H. Wang, H.-J. Huang, J.-T. Lin, C.-W. Hu, K.-H. Zeng, and M. Sun, "Omnidirectional cnn for visual place recognition and navigation," in *In Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, 2018.

[45] A. Zamir, F Xia, J He, A Sax, J Malik, and S Savarese, "Gibson env: Real-world perception for embodied agents," in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 4, 2018.

[46] H. Yu, X. Lian, H. Zhang, and W. Xu, "Guided feature transformation (gft): A neural language grounding module for embodied agents," *arXiv preprint arXiv:1805.08329*, 2018.

[47] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[48] Y. Xu, A. Fern, and S. W. Yoon, "Discriminative learning of beam-search heuristics for planning.," in *IJCAI*, 2007, pp. 2041–2046.

[49] C. M. Wilt and W. Ruml, "Building a heuristic for greedy search," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[50] M. Bhardwaj, S. Choudhury, and S. Scherer, "Learning heuristic search via imitation," *arXiv preprint arXiv:1707.03034*, 2017.

[51] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[52] C.-Y. Ma, Z. Wu, G. AlRegib, C. Xiong, and Z. Kira, "The regretful agent: Heuristic-aided navigation through progress estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[53] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine, "Leave no trace: Learning to reset for safe and autonomous reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[54] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1933–1941.

[55] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[56] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[57] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1080–1088.

[58] Y. Li, C. Huang, C. C. Loy, and X. Tang, "Human attribute recognition by deep hierarchical contexts," in *European Conference on Computer Vision*, Springer, 2016, pp. 684–700.

[59] X. Peng and C. Schmid, "Multi-region two-stream r-cnn for action detection," in *European Conference on Computer Vision*, Springer, 2016, pp. 744–759.

[60] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, "Jointly modeling embedding and translation to bridge video and language," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4594–4602.

[61] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4534–4542.

[62] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2014, pp. 1495–1504.

[63] V. Ramanishka, A. Das, J. Zhang, and K. Saenko, "Top-down visual saliency guided by captions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[64] J. Song, Z. Guo, L. Gao, W. Liu, D. Zhang, and H. T. Shen, "Hierarchical lstm with adjusted temporal attention for video captioning," *International Joint Conference on Artificial Intelligence*, 2017.

[65] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4507–4515.

[66] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, "Video paragraph captioning using hierarchical recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4584–4593.

[67] M. Zanfir, E. Marinoiu, and C. Sminchisescu, "Spatio-temporal attention models for grounded video captioning," in *Asian Conference on Computer Vision*, Springer, 2016, pp. 104–119.

[68] Z. Gan, C. Gan, X. He, Y. Pu, K. Tran, J. Gao, L. Carin, and L. Deng, "Semantic compositional networks for visual captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[69] Y. Pan, T. Yao, H. Li, and T. Mei, "Video captioning with transferred semantic attributes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[70] Z. Shen, J. Li, Z. Su, M. Li, Y. Chen, Y.-G. Jiang, and X. Xue, "Weakly supervised dense video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[71] Y Yu, H Ko, J Choi, and G Kim, "End-to-end concept word detection for video captioning, retrieval, and question answering," in *CVPR*, vol. 3, 2017, p. 7.

[72] Y.-W. Chao, Y. Liu, X. Liu, H. Zeng, and J. Deng, "Learning to detect human-object interactions," *arXiv preprint arXiv:1702.05448*, 2017.

[73] G. Gkioxari, R. Girshick, P. Dollár, and K. He, "Detecting and recognizing human-object interactions," *arXiv preprint arXiv:1704.07333*, 2017.

[74] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3668–3678.

[75] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene graph generation from objects, phrases and region captions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1261–1270.

[76] X. Liang, L. Lee, and E. P. Xing, "Deep variation-structured reinforcement learning for visual relationship and attribute detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[77] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[78] B. Dai, Y. Zhang, and D. Lin, "Detecting visual relationships with deep relational networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[79] R. Hu, M. Rohrbach, J. Andreas, T. Darrell, and K. Saenko, "Modeling relationships in referential expressions with compositional modular networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[80] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in Neural Information Processing Systems*, 2017.

[81] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, "Visual translation embedding network for visual relation detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[82] J. Zhang, M. Elhoseiny, S. Cohen, W. Chang, and A. Elgammal, "Relationship proposal networks," in *CVPR*, vol. 1, 2017, p. 2.

[83] B. Ni, V. R. Paramathayalan, and P. Moulin, "Multiple granularity analysis for fine-grained action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 756–763.

[84] B. Ni, X. Yang, and S. Gao, "Progressively parsing interactional objects for fine grained action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1020–1028.

[85] C. Lea, A. Reiter, R. Vidal, and G. D. Hager, "Segmental spatiotemporal cnns for fine-grained action segmentation," in *European Conference on Computer Vision*, Springer, 2016, pp. 36–52.

[86] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 6.

[87] J. Lu, J. Yang, D. Batra, and D. Parikh, "Neural baby talk," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7219–7228.

[88] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2625–2634.

[89] A. Rohrbach, A. Torabi, M. Rohrbach, N. Tandon, C. Pal, H. Larochelle, A. Courville, and B. Schiele, "Movie description," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 94–120, 2017.

[90] S. Venugopalan, L. Anne Hendricks, M. Rohrbach, R. Mooney, T. Darrell, and K. Saenko, "Captioning images with diverse objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5753–5761.

[91] A. Rohrbach, M. Rohrbach, S. Tang, S. Joon Oh, and B. Schiele, "Generating descriptions with grounded and co-referenced people," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4979–4989.

[92] R. Shetty, M. Rohrbach, L. Anne Hendricks, M. Fritz, and B. Schiele, "Speaking the same language: Matching machine to human captions by adversarial training," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4135–4144.

[93] J. S. Park, M. Rohrbach, T. Darrell, and A. Rohrbach, "Adversarial inference for multi-sentence video description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[94] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 3156–3164.

[95] Y. Li, T. Yao, Y. Pan, H. Chao, and T. Mei, "Jointly localizing and describing events for dense video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7492–7500.

[96] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei, "Boosting image captioning with attributes," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 22–29.

[97] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4651–4659.

[98] L. Zhou, C. Xu, P. Koch, and J. J. Corso, "Watch what you just said: Image captioning with text-conditional attention," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, ACM, 2017, pp. 305–313.

[99] M. Zanfir, E. Marinoiu, and C. Sminchisescu, "Spatio-temporal attention models for grounded video captioning," in *Asian Conference on Computer Vision*, 2016, pp. 104–119.

[100] P. Das, C. Xu, R. F. Doell, and J. J. Corso, "A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2634–2641.

[101] G. Kulkarni, V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Babytalk: Understanding and generating simple image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2891–2903, 2013.

[102] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[103] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, IEEE, 2017, pp. 2980–2988.

[104] A. Das, H. Agrawal, L. Zitnick, D. Parikh, and D. Batra, "Human attention in visual question answering: Do humans and deep networks look at the same regions?" *Computer Vision and Image Understanding*, vol. 163, pp. 90–100, 2017.

[105] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, "Grounding of textual phrases in images by reconstruction," in *European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 817–834.

[106] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, "Natural language object retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4555–4564.

[107] F. Xiao, L. Sigal, and Y. Jae Lee, "Weakly-supervised visual grounding of phrases with linguistic structures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5945–5954.

[108] C. Deng, Q. Wu, Q. Wu, F. Hu, F. Lyu, and M. Tan, "Visual grounding via accumulated attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7746–7755.

[109] L. Zhou, Y. Kalantidis, X. Chen, J. J. Corso, and M. Rohrbach, "Grounded video description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[110] Y. Zhang, J. C. Niebles, and A. Soto, "Interpretable visual question answering by visual grounding from attention supervision mining," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2019, pp. 349–357.

[111] L. Anne Hendricks, R. Hu, T. Darrell, and Z. Akata, "Grounding visual explanations," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 264–279.

[112] R. R. Selvaraju, S. Lee, Y. Shen, H. Jin, D. Batra, and D. Parikh, "Taking a hint: Leveraging explanations to make vision and language models more grounded," *arXiv preprint arXiv:1902.03751*, 2019.

[113] F. Wang, Q. Huang, and L. J. Guibas, "Image co-segmentation via consistent functional maps," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

[114] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2223–2232.

[115] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma, "Dual learning for machine translation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 820–828.

[116] X. Chen and C Lawrence Zitnick, "Mind's eye: A recurrent visual representation for image caption generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 2422–2431.

[117] M. Shah, X. Chen, M. Rohrbach, and D. Parikh, "Cycle-consistency for robust visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[118] D. Tang, N. Duan, Z. Yan, Z. Zhang, Y. Sun, S. Liu, Y. Lv, and M. Zhou, "Learning to collaborate for question answering and asking," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, vol. 1, 2018, pp. 1564–1574.

[119] B. Wang, L. Ma, W. Zhang, and W. Liu, "Reconstruction network for video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7622–7631.

[120] X. Duan, W. Huang, C. Gan, J. Wang, W. Zhu, and J. Huang, "Weakly supervised dense event captioning in videos," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 3063–3073.

[121] Q. Huang, P. Zhang, D. Wu, and L. Zhang, "Turbo learning for captionbot and drawingbot," in *Advances in Neural Information Processing Systems (NeurIPS*, 2018, pp. 6456–6466.

[122] E. Hosseini-Asl, Y. Zhou, C. Xiong, and R. Socher, "Augmented cyclic adversarial learning for low resource domain adaptation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[123] Y. Benn, T. L. Webb, B. P. Chang, Y.-H. Sun, I. D. Wilkinson, and T. F. Farrow, "The neural basis of monitoring goal progress," *Frontiers in human neuroscience*, vol. 8, p. 688, 2014.

[124] C. H. Chatham, E. D. Claus, A. Kim, T. Curran, M. T. Banich, and Y. Munakata, "Cognitive control reflects context monitoring, not motoric stopping, in response inhibition," *PloS one*, vol. 7, no. 2, e31546, 2012.

[125] E. T. Berkman and M. D. Lieberman, "The neuroscience of goal pursuit," *The psychology of goals*, pp. 98–126, 2009.

[126] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.

[127] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[128] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[129] R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and executing generalized robot plans," *Artificial intelligence*, vol. 3, pp. 251–288, 1972.

[130] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[131] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," in *European Conference on Computer Vision (ECCV)*, Springer, 2016, pp. 510–526.

[132] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[133] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles, "Dense-captioning events in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[134] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," in *CRCV-TR-12-01*, 2012.

[135] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: A large video database for human motion recognition," in *2011 International Conference on Computer Vision (ICCV)*, IEEE, 2011, pp. 2556–2563.

[136] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1725–1732.

[137] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah, "The thumos challenge on action recognition for videos "in the wild"," *Computer Vision and Image Understanding*, vol. 155, pp. 1–23, 2017.

[138] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 961–970.

[139] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[140] C.-Y. Ma, M.-H. Chen, Z. Kira, and G. AlRegib, "Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition," *arXiv preprint arXiv:1703.10667*, 2017.

[141] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and vqa," *arXiv preprint arXiv:1707.07998*, 2017.

[142] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[143] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[144] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[145] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European Conference on Computer Vision*, Springer, 2016, pp. 20–36.

[146] B. Ghanem, J. C. Niebles, C. Snoek, F. C. Heilbron, H. Alwassel, R. Khrisna, V. Escorcia, K. Hata, and S. Buch, "Activitynet challenge 2017 summary," *arXiv preprint arXiv:1710.08011*, 2017.

[147] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[148] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, Barcelona, Spain, vol. 8, 2004.

[149] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, vol. 29, 2005, pp. 65–72.

[150] R. Vedantam, C Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[151] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via a visual sentinel for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[152] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2641–2649.

[153] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European Conference on Computer Vision*, Springer, 2016, pp. 382–398.

[154] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.

[155] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[156] L. Zhou, N. Louis, and J. J. Corso, "Weakly-supervised video object grounding from text by loss weighting and object interaction," in *British Machine Vision Conference (BMVC)*, 2018.

[157] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta, "Pyrobot: An open-source robotics framework for research and benchmarking," *arXiv preprint arXiv:1906.08236*, 2019.

[158] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[159] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*, IEEE, 2018.

[160] F. Xia, C. Li, K. Chen, W. B. Shen, R. Martín-Martín, N. Hirose, A. R. Zamir, L. Fei-Fei, and S. Savarese, "Gibson env v2: Embodied simulation environments for interactive navigation," Stanford University, Tech. Rep., Jun. 2019.

[161] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2556–2565.

[162] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[163] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.